"LEARNING = INTERPRETATION + GENERALIZATION:"
A CASE STUDY IN KNOWLEDGE-DIRECTED LEARNING

Elliot M. Soloway

COINS Technical Report 78-13

(July 1978)

This work was supported by the United States Army Research Institute for the Behavioral and Social Sciences under Grant Numbers DAHC19-77-G-0012 and DAHC19-76-G-0013.

"LEARNING = INTERPRETATION + GENERALIZATION:" A CASE STUDY IN KNOWLEDGE-DIRECTED LEARNING

A Dissertation Presented

By

ELLIOT MORRIS SOLOWAY

Submitted to the Graduate School of the
University of Massachusetts in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

September 1978

Computer and Information Science

This work was supported by

United States Army Research Institute
for the Behavioral and Social Sciences

under Grant Numbers

DAHC19-77-G-0012
DAHC19-76-G-0013

"LEARNING = INTERPRETATION + GENERALIZATION:"
A CASE STUDY IN KNOWLEDGE-DIRECTED LEARNING

A Dissertation Presented

By

ELLIOT MORRIS SOLOWAY

Approved as to style and content by:

Edward M. Riseman

Edward M. Riseman, Chairperson of Committee

M. A. Arbib

Michael A. Arbib, Member

Allen R. Hanson

Allen R. Hanson, Member

William Kilmer

William L. Kilmer, Member

Robert M Graham

Robert M. Graham, Department Head
Computer and Information Science

To my brother, Eddie

ACKNOWLEDGEMENTS

While I have written numerous drafts of this page in my mind over the past three years, words now fail me. How can one acknowledge all those networks of individuals who contributed -- knowingly or not -- to a dissertation? But, I must try.

It is with the most sincere gratitude that can be mustered, that I thank

> Dr. Edward Riseman, chairman of my committee, who is truly a good human being; without his unflagging emotional and intellectual support this work would never have been done,
>
> Dr. Michael Arbib, committee member, whose very presence is stimulating; he creates an environment in which these difficult research questions can be explored, and helps provide a perspective on the important and the not so important,
>
> Dr. Allen Hanson, committee member, whose quiet competence -- and backstage maneuvering -- helped in my eleventh hour push,
>
> Dr. William Kilmer, committee member, who embodies the finest tradition of scientific inquiry, and with whom I have spent many enjoyable hours discussing all sorts of questions.

I would also like to thank the following individuals in the field, who made technical and personal contributions to this work:

> Ed Fisher, Victor Lesser, John Lowrance, Jon Post, Robbie Moll, Nils Nilsson, Chuck Schmidt, Sridharan, Jim Stanley, and Edwin Williams.

Let me also nominate to the hall of saints Janet Turnbull and Donna Jerkovich; their administrative, secretarial and social work skills are greatly appreciated.

To those who ministered to the needs of my body and soul, thank you Drs. Barry Slater, Doris Summerson, Tony Melchionda, and Tom Wolff.

The staff of the Computing Center at the University of Massachusetts were most generous with their time -- and that of the computer, the

support of Dr. Conrad Wogrin, Chuck Lyman, Bud Maziarz, Paul McElroy, Donn Lord, Joe Hicks, Arthur Mann, is much appreciated.

And finally, to my non-computer friends (sic) whose questions, general disbelief and good humor provided a welcome and healthy outside perspective, thank you

> Sandy, Vance, Sherri, Sanford, Jack, Annie, Stephen, California Barry, Takeh Barry, Jeanie, John, Jane, Paul, Jude, and Judy for constantly reminding me that it was all a "process" and making me take Stresstabs everyday, and my wonderful mother, who, rightfully so, still doesn't believe I will get a job.

ABSTRACT

"Learning = Interpretation + Generalization:"
A Case Study in Knowledge-Directed Learning

September 1978

Elliot Soloway, B.A., Ohio State University

M.S., Ph.D., University of Massachusetts

Directed by: Professor Edward Riseman

The crucial role which domain knowledge plays in the interpretation and generalization phases of the learning process is explored. The setting for this work is the competitive action game of baseball; a system is described which acquires rules that relate the observed actions of the players to their goals in the game. Multiple levels of knowledge and processing are used to transform the input descriptions of physical activity (e.g., RUN, CATCH, THROW) into the desired generalized interpretations (e.g., "hits", "outs").

The system is provided initially with general knowledge about the types of goals and relationships often found in competitive action games, and common-sense knowledge about the properties and relationships of physical actions. This knowledge is used in various levels of interpretation to filter out irrelevant actions in the input, link actions together by inferring causal relationships, and attribute purposes to the players by hypothesizing competitive and cooperative goals for their actions.

Generalization results in the formation of classes of events (rules) at various levels of abstraction. We critically examine both the data-directed (DDG) and the knowledge-directed (KDG) approaches to generalization. The conclusion we draw is that domain knowledge must enter the learning loop in order for a system to learn. In systems which employ DDG, that knowledge is supplied by some external source, whereas in our KDG system, that knowledge has been made explicit and provided to the system itself.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

I.1 The Need for Domain Knowledge in Learning

The nature of "learning" has been debated throughout the centuries. Plato posed the philosophical problem of how one can come to know most eloquently in his famous Paradox of the Meno [JOW49]:

> How could the slave boy be taught the proof of the Pythagorean Theorem? If the boy did not know it already then how could he recognize it, and if he knew it already then certainly the boy was not learning it.

From the perspective of computers, the challenge of learning could be characterized as

> How could a machine do something that it was not originally programmed to do?

A tension expressed in these two statements can be summarized as follows:

> What knowledge must a system bring to a situation in order to learn?
>
> vs.
>
> What knowledge is "fair" for a system to bring to a situation in order to learn?

The issue of "what knowledge" or "how much knowledge" is of considerable importance to the engineering of artificially intelligent systems. It has been argued, in fact, that in order to develop high performance systems, a significant amount of knowledge about the task domain must be employed [FEI77]. Expert problem solving systems have been built using this philosophy (e.g., [SHO76, DUD77]). In such systems it is perfectly

"fair" -- in fact encouraged -- to include as much domain knowledge as possible. However, in building a system which is supposed to learn, there is legitimate concern for the appropriateness of including domain knowledge. Clearly, if too much domain knowledge is included in the system, the learning task is trivialized. On the other hand, is it possible to build a learning system which does not need to employ domain knowledge and which can operate in any problem domain (cf., [VER75])?

In this thesis, we describe our attempt to reconcile the need for domain knowledge with the requirements of fairness imposed on that knowledge by the learning task. The conclusion we draw is that domain knowledge must enter the learning loop (observations, teacher, learning system) somewhere, if that system is to learn. This is a strong claim; in systems which are purportedly domain independent, we shall argue that the critical domain knowledge is embodied in either an explicit teacher, who provides classification and other training information, or an implicit teacher, who structures and defines the observations. In our work, we have equipped the system itself with knowledge of the problem domain in order to more carefully study the contribution of domain knowledge to the learning process.

The task domain in which we shall explore these issues is the competitive action game of baseball. In particular, the goal of the learning program described here is to acquire an understanding of some of the more important relationships between the actions of players and their goals in the American game of baseball.[1] As simulated observations of

events in baseball are input, the system will acquire rules which describe complex concepts such as 'hit' and 'out'. It is also imperative that the system be able to use the knowledge which it has acquired; as subsequent events are observed the system will recognize new instances and hypothesize new relationships using the acquired rules.

In this chapter we shall first present an outline of the contents of the other chapters in this thesis and suggest a strategy for its reading. Next, an annotated example of the system's performance is presented. This will be followed by a discussion of the model of learning employed by the system. We close this chapter with a list of questions which will be addressed in subsequent chapters.

I.2 Outline of Chapters

The organization of the thesis is as follows:

Chapter II provides a brief overview of the major aspects of the system.

Chapter III describes the content and representation of the knowledge concerning spatio-temporal activity with which the system is equipped.

Chapter IV discusses a number of issues: the content and representation of the knowledge initially provided to the system concerning competition and cooperation in games; an evaluation of the contribution of that knowledge to the learning process; various issues surrounding the use of acquired knowledge; a survey of other systems which exhibit interpretive behavior.

Chapter V describes the control structure which oversees the interpretation process.

Chapter VI is another multi-issue chapter; a survey and critical analysis of systems which employ the data-directed approach to generalization is presented, followed by a survey of systems which employ knowledge-directed generalization; a discussion of our system's technique is included here.

Chapter VII describes how our system gathers and evaluates evidence in order to determine the correctness or incorrectness of its hypothesized interpretations.

Chapter VIII reports on a series of experiments performed with the system.

Chapter IX provides a summary of the issues raised in the thesis.

Taken together, Chapters I and II are intended to provide a summary of issues discussed in detail in subsequent chapters; the reader might well read them plus the conclusions chapter (IX) before plunging into the detailed discussions.

I.3 An Annotated Example from Our Learning System

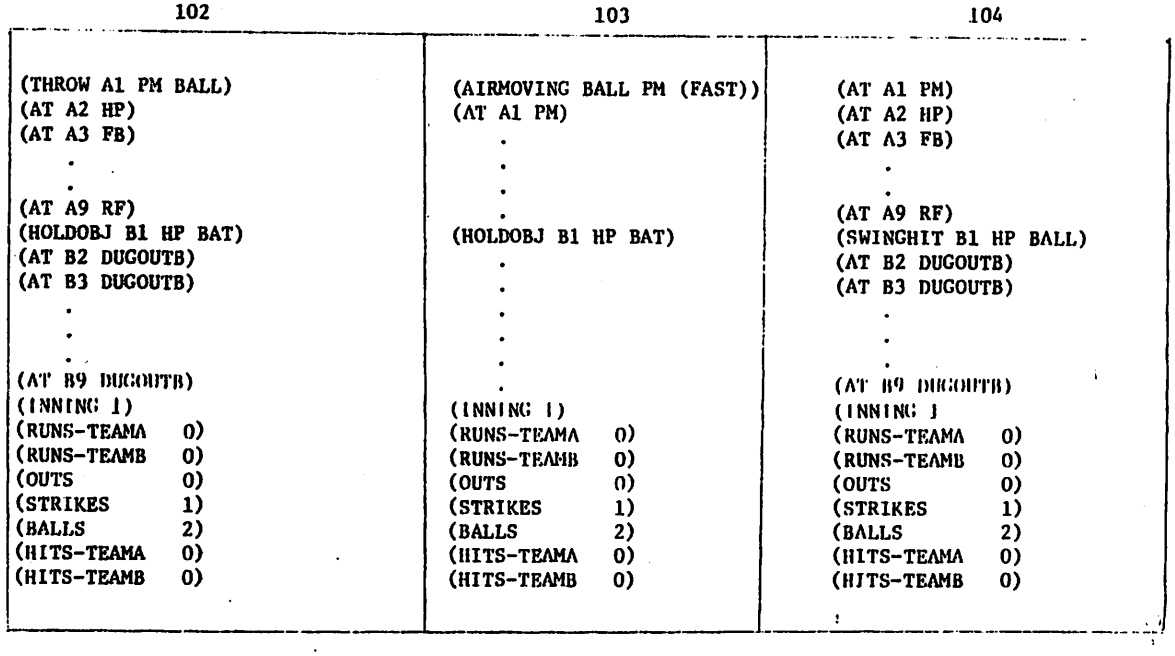
The following example is intended to convey to the reader a feeling for the types of processes and knowledge that are used in the task at hand.

I.3.1 Focussing Attention on Interesting Observations and Segmenting Continuous Activity into Episodes

To simulate the actual viewing of a baseball game for the benefit of our presently blind computer, we have broken the continuous stream of activity in that game into discrete time slices called snapshots. In games generated by a FORTRAN program, there are approximately 2000

snapshots per game. Figure I.1 depicts three such snapshots. In snapshot 102 (Figure I.1) we see a player, A1, THROWing a BALL from the PITCHER'S-MOUND, while his other teammates, A2 through A9, are standing AT their respective positions. We also see an opposing player, B1, standing at HOMEPLATE HOLDing a stick-like OBJECT called a BAT, while his teammates are AT a designated waiting location called DUGOUTB. Snapshot 103 (Figure I.1) depicts the BALL MOVING FAST through the AIR. Finally, in snapshot 104, we see player B1 SWINGing the BAT and HITting the BALL, which was THROWn by the pitcher A1. The set of features describing a player's action will be called a pattern description. Note, as far as the system is concerned PITCHER'S-MOUND, HOMEPLATE, etc. are simply X-Y spatial coordinates -- they have no a priori significance. The machine learns that some locations are in fact important, e.g., FIRSTBASE. The system does have a priori knowledge, however, that there are two teams playing the game (the A team and the B team), and it also does know the team affiliation of each individual player in view.

The description of the activity that is input to the system includes only "perceptual features" such as action, actor, location, and time-of-occurrence. These descriptions do not include higher level features[2] such as the "goals" of the player or "relationships" between players; these later features must be inferred by the system and added to the descriptions. It is crucial for the reader to note here that the input descriptions do not contain all the important features.



The pitcher throws the ball.

The ball moves through the air.

The batter hits the ball towards the shortstop.

Figure I.1 Typical Unfiltered Snapshots

Each snapshot describes the activity of each player at a moment in time. Time is encoded implicitly in the sequencing of the snapshots. Symbolic names for locations, such as PM (pitcher's mound), are used for the reader's convenience; the system knows them only in terms of their X-Y spatial coordinates. HP stands for homeplate, FB for firstbase, RF for right field.

Before leaving Figure I.1, notice that most of the activity of the players is unchanging; e.g., player A3 remains standing AT FIRSTBASE throughout the three depicted snapshots. Since the volume of information flowing into the system is great, it is useful to reduce that data to a more manageable size. If we focus our attention only on movement or changes in action, we would in fact not notice A3, but rather would see A1 THROWing the BALL, the BALL MOVing, and B1 HITting the BALL. Our system uses this biologically-motivated heuristic as a crude first-pass filter to eliminate all nonchanging activity. Figure I.2 depicts the result of applying this heuristic to the snapshots in Figure I.1. While nonchanging activity may sometimes actually be important, we shall rely on later processing to redirect the system's attention to such nonchanging activity when necessary.

Not all the activity in a game is significant. Some actions might be described as "ritualistic", a batter WALKing to HOMEPLATE or a fielder returning the BALL to the pitcher, etc. In order to highlight the important sequences of competitive interactions, the continuous stream of snapshots is segmented into units call episodes. Typical episodes would be an infield single, or a groundout. This segmentation is based on the observation that the action sequences of interest -- the ones in which competitive interactions take place -- are often indicated by a high degree of activity surrounded by low activity sequences. For example, prior to the batter HITting the BALL (an example of a high degree activity), the pitcher was simply standing AT the PITCHER'S-MOUND HOLDing the BALL (an example of low degree activity). After the batter has RUN to FIRST-

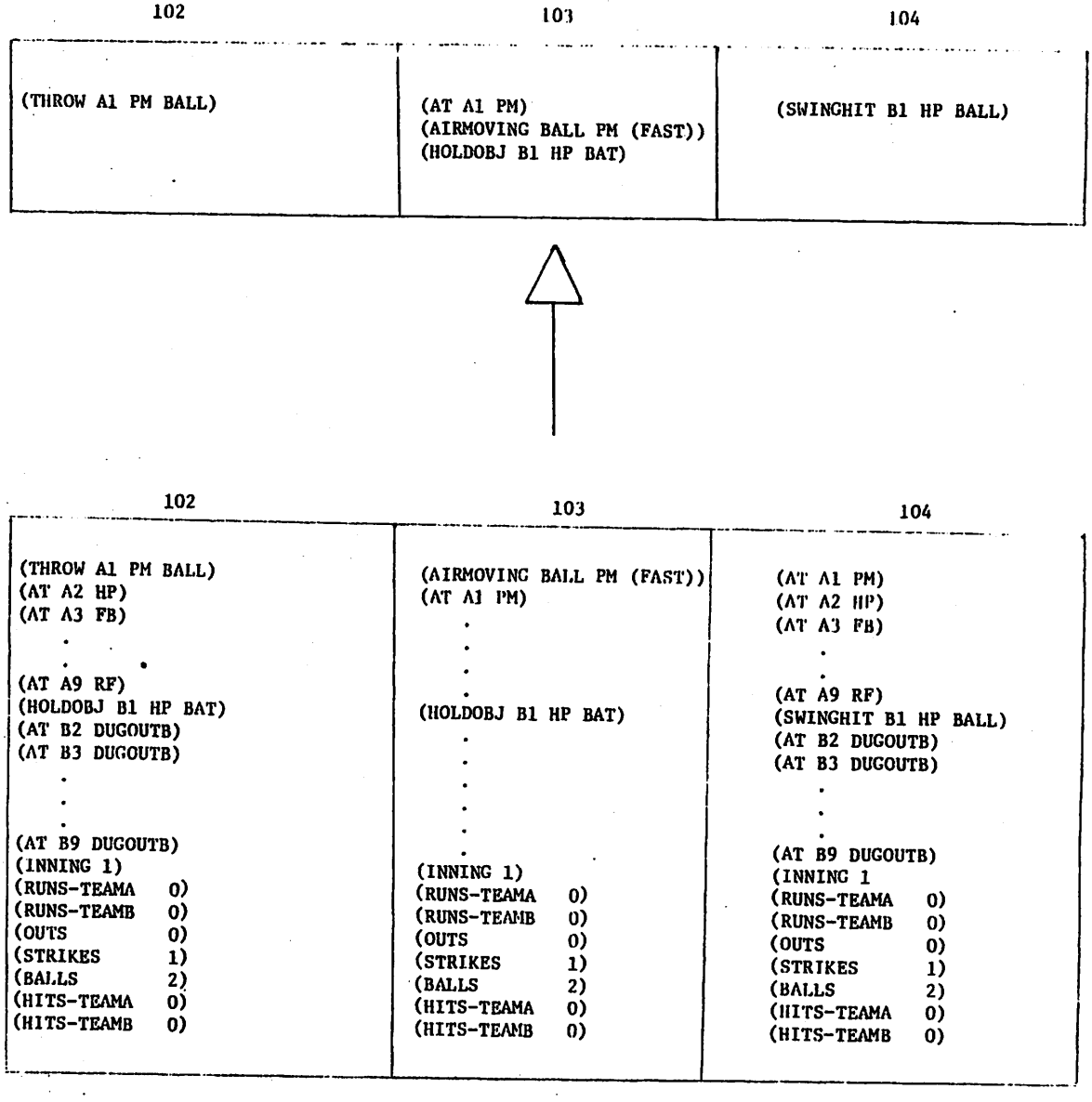


Figure I.2 Filtering Out Apparently Irrelevant Activity

The lower half of this figure reproduces the input depicted in Figure I.1, while the upper half of this figure displays the effect of applying the heuristic "filter out non-changing activity" to that input. Note the drastic reduction in the number of events in a snapshot.

BASE or back to his DUGOUT, the pitcher is again HOLDing the BALL. Such a sequence of snapshots would constitute an episode.

I.3.2 First Level of Interpretation: Using a Model of the Common-Sense Physics of Actions

When we see player B1 HIT the BALL, we know that it was the THROW by player A1 which set up some of the physical conditions necessary for B1's subsequent action, i.e., A1's THROW put the BALL into the AIR. We equip our learning system with a similar common-sense understanding of the physical relationships of actions. Using this knowledge, our system infers a causal relationship between the THROW of A1 and the SWINGHIT of B1; that is, A1 enabled B1 to HIT the BALL (Figure I.3). Note that connecting the actions of the players in this way is still independent of the game of baseball. Finally, Figure I.4 shows how the system expresses this inferred relationship. A new "feature" has been added to the pattern descriptions of the participating actions which indicates the inferred causal relationship.

I.3.3 Second Level of Interpretation: Inferring Goals Using a Model of Competitive Games

In order to understand the descriptions of activity as a game of baseball, we must interpret those descriptions in light of a general understanding of games. If a general understanding of religion were used to view the same activity, one might well interpret that activity as some religious ceremony! In other words, in order to "see" the activity as a game of baseball, the observer -- be it man or machine -- requires a model of competitive games.

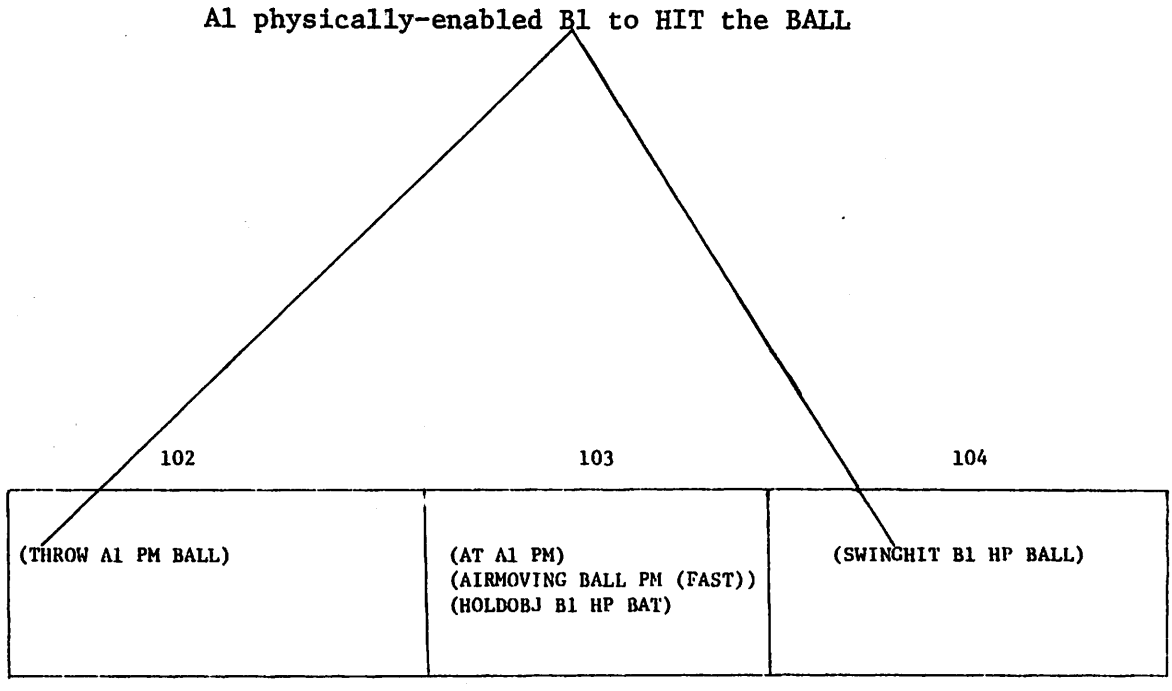


Figure I.3 Inferring Relationships Between Actions

On the basis of its common-sense understanding of the physical relationships between actions, the system inferred a causal relationship between the THROW action of player A1 and the SWINGHIT action of player B1. Namely, A1's action set up the enabling conditions for B1's subsequent action.

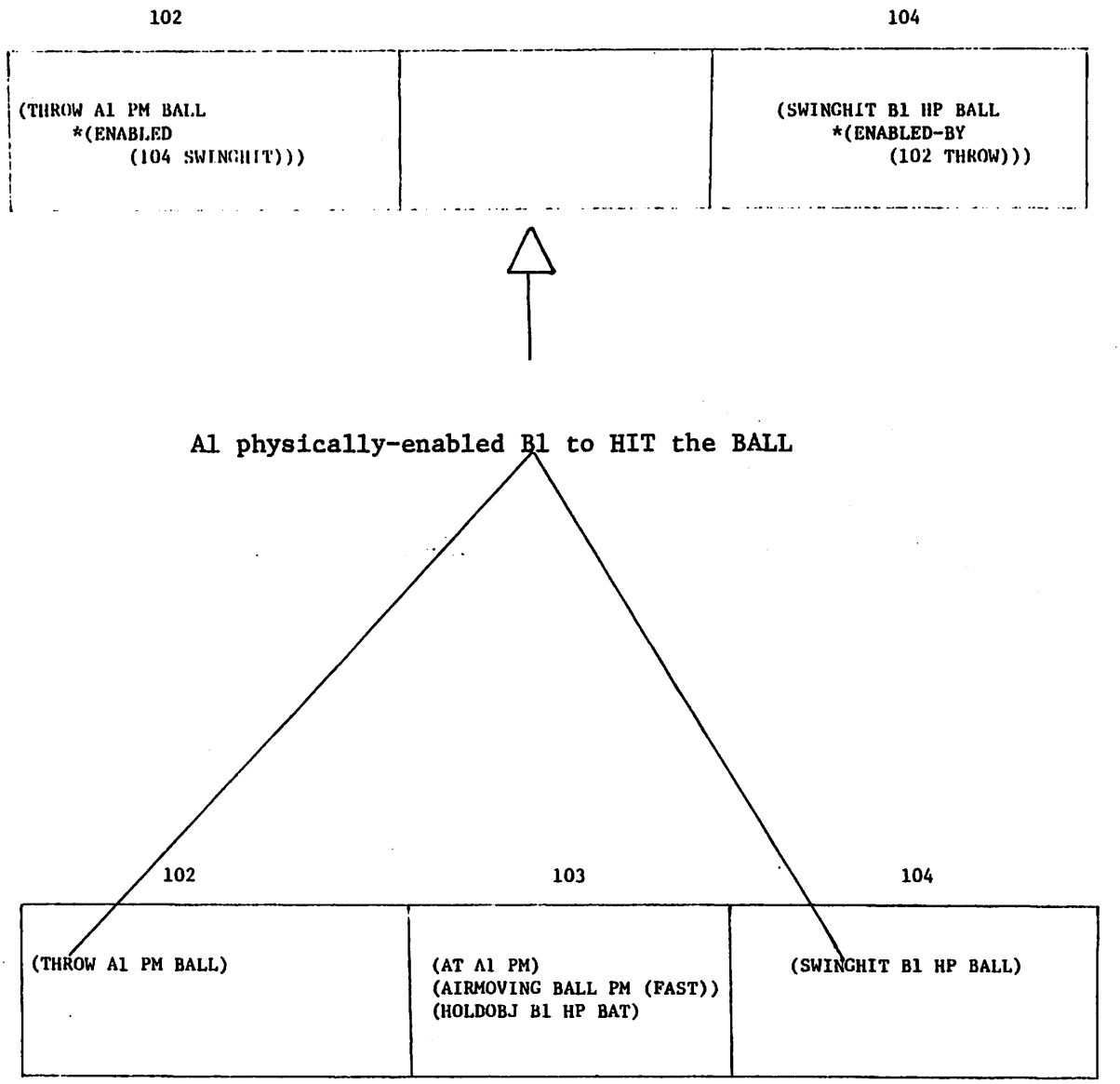


Figure I.4 Adding a New Feature to the Description of Activity

This figure indicates how a new feature (indicated by an asterisk) can be added to the activity description based on the inference of the relationship between players A1 and B1. Note, this relationship is independent of games; it simply reflects our common-sense understanding of actions.

The following general fact would probably be in such a model of competitive action games: a competitive interaction is probably occurring when it is observed that one player's actions enable an opposing player to execute an action which requires a measure of skill and/or energy. Using this fact in the example snapshots from Figure I.4, we can hypothesize that the pitcher A1 was in competition with the opposing player B1, since we had previously inferred that A1 physically enabled B1 to execute the action SWINGHIT and we know that HITting a BALL which was moving FAST does require a high degree of skill (Figure I.5).

Other general facts about competition would permit a system to hypothesize goals for the players involved in competitive interactions. For example, the following fact could be used to hypothesize goals for the competitive interaction between the pitcher A1 and the opposing batter B1 (Figure I.5): a player who enabled an opponent's action might actually have been trying to prevent the opponent's action, while the opponent had indeed wanted to execute the observed action.[3] Thus, the pitcher A1 was actually trying to THROW the BALL towards the location where the batter B1 was standing (HOMEPLATE, Figure I.5) in such a way as to prevent B1 from being able to execute the action of HITting the BALL, while still providing B1 with a reasonable opportunity to hit it. Since the batter B1 did HIT the BALL in the observed snapshots, we would say that in the observed competitive interaction the pitcher failed in his goal of preventing his opponent from HITting the BALL. Moreover, since HITting a BALL which is traveling fast requires a high degree of

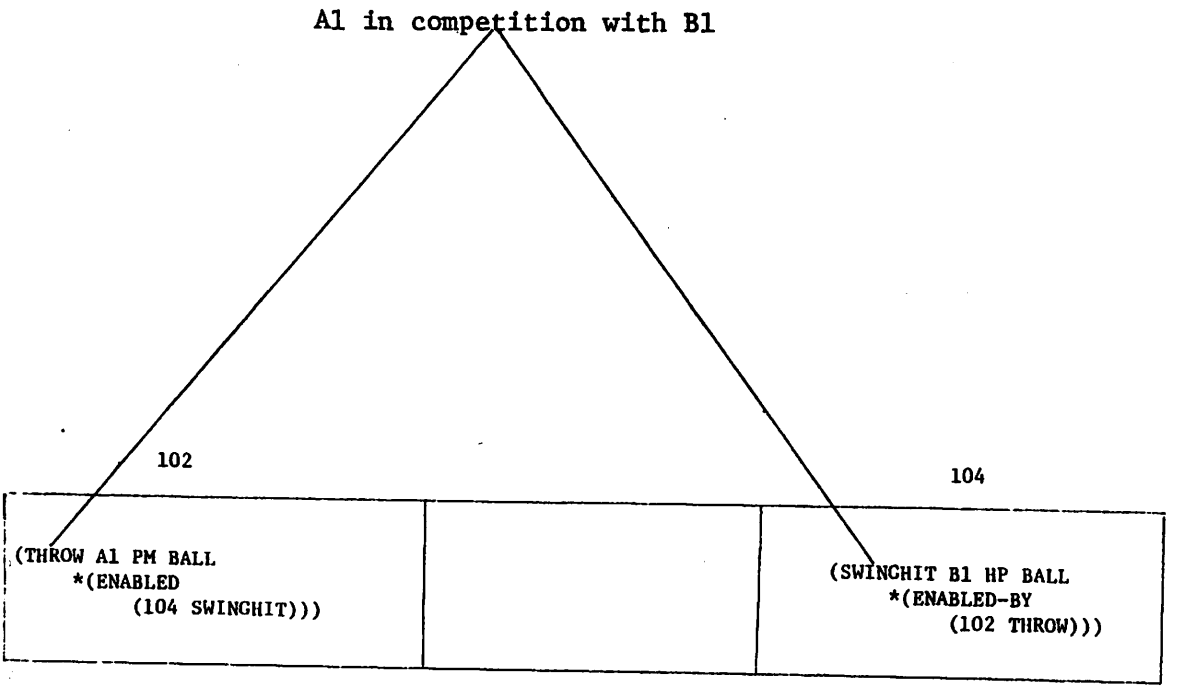


Figure I.5 Understanding the Observed Activity as a Competitive Game

General knowledge about games must be brought to bear in order to interpret the observed activity as a particular game. The fact that in games a player does not intend to enable an opponent's action, allows the system to hypothesize that A1 and B1 are involved in a competitive interaction.

skill and energy, we can hypothesize that the batter was trying to execute the observed action and thus he succeeded with his goal in the observed instance. We must emphasize that these inferences are hypotheses which may be in error. Subsequent processing will gather evidence for the correctness (or incorrectness) of such hypotheses.

Facts of the above sort constitute the general model of competitive action games which is initially provided to the system. Employing such a model, the system's final analysis of the snapshots, depicted in Figure I.6, is the further augmentation of the descriptions of the observed activity. A similar analysis of the snapshots in a whole episode results in the hypothesis of a number of competitive and cooperative interactions; Figure I.7 illustrates the set of competitive and cooperative interactions discovered in the analysis of an 'infield single'. Now the induction of general rules from specific instances can take place.

I.3.4 Generalization: Constructing a Hierarchy of General Classes Based on Similarities of Descriptions

The hypotheses which are finally verified as correct will be used by the system to analyze subsequently observed activity, i.e., infer goals and relationships for events in baseball. The competitive interaction hypothesized above between the pitcher and the batter is an example of such an eventual rule. However, as it is stated in Figure I.6, it is too specific -- it refers only to players A1 and B1 at time 102 and 104. During Hypothesis Generalization specific hypotheses, such

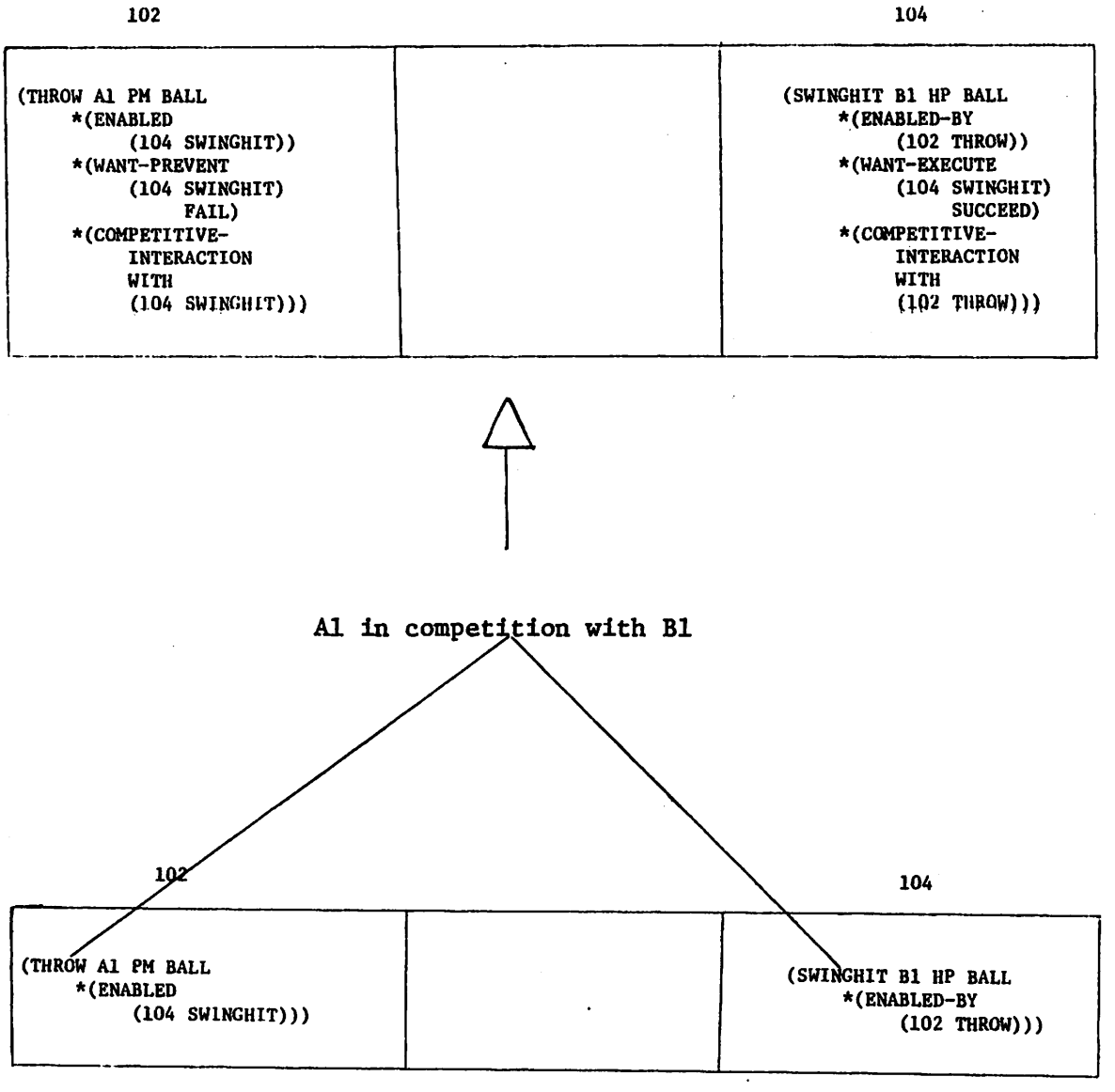


Figure I.6 Adding New Features to the Description of the Activity

The system attempts to infer the goals of the players involved in a competitive interaction. In the above case, it is hypothesized that the goal of player A1 was to prevent B1 from HITting the BALL, while the goal of player B1 succeeded with his intention to HIT the BALL; this inference follows directly from the nature of a competitive interaction.

```
(101  THROW A1 PM BALL                    (103  SWINGHIT B1 HP BALL
      (ENABLED (103 SWINGHIT))                  (ENABLED-BY (101 THROW))
      (WANT PREVENT                             (WANT EXECUTE
       (103 SWINGHIT) FAIL))                     (103 SWINGHIT) SUCCEED))
      (PHYSICAL-COMPETITION                     (PHYSICAL-COMPETITION
        WITH (103 SWINGHIT))                      WITH (101 THROW))
       DIFFICULT-ACT                             DIFFICULT-ACT
      (CAN-EFFECT-PERFORMANCE                   CHANGED-ACT)
       (103 SWINGHIT)))
```

```
(103  SWINGHIT B1 HP BALL                 (106  CATCH A4 SB BALL
      (ENABLED (106 CATCH))                     (ENABLED-BY (103 SWINGHIT))
      (WANT PREVENT                             (WANT EXECUTE
       (106 CATCH) FAIL)                         (106 CATCH) SUCCEED)
      (PHYSICAL-COMPETITION                     (PHYSICAL-COMPETITION
        WITH (106 CATCH))                         WITH (103 SWINGHIT))
       DIFFICULT-ACT                             DIFFICULT-ACT
      (CAN-EFFECT-PERFORMANCE                   CHANGED-ACT)
       (106 CATCH)))
```

```
(110  CATCH A3 FB BALL                    (109  ON B1 FB
      (WANT PREVENT                             (WANT EXECUTE
       (109 ON) FAIL)                            (109 ON) SUCCEED)
      (ORDER-OF-OCCURRENCE-                     (ORDER-OF-OCCURRENCE-
       COMPETITION WITH (109 ON))                COMPETITION WITH (110 CATCH))
       DIFFICULT-ACT                             NOT-DIFFICULT-ACT
      (OCCURS-AFTER                             (OCCURS-BEFORE
       (109 ON)))                                (110 CATCH)))
```

```
(103  SWINGHIT B1 HP BALL                 (104  RUN B1 HP (FAST)
      (ENABLED(104 RUN))                        (ENABLED-BY
      (WANT EXECUTE                               (103 SWINGHIT))
        (103 SWINGHIT) SUCCEED)                 (WANT EXECUTE
      (ORDER-OF-OCCURRENCE-                       (103 RUN) SUCCEED)
         COOPERATION WITH                       (ORDER-OF-OCCURRENCE-
        (104 RUN)))                                COOPERATION WITH
                                                   (103 SWINGHIT))
```

```
(107  THROW A4 SB BALL                    (110  CATCH A3 FB BALL
      (ENABLED (109 CATCH))                     (ENABLED-BY
      (WANT EXECUTE                               (107 THROW))
        (103 CATCH) SUCCEED)                    (WANT EXECUTE
      (PHYSICAL-COOPERATION WITH                  (110 CATCH) SUCCEED)
        (110 CATCH)))                           (PHYSICAL-COOPERATION WITH
                                                   (107 THROW)))
```

Figure I.7 Competitive and Cooperative Interactions in an Infield Single Episode

as the one in Figure I.6, are given a more general characterization. In particular, this process has two objectives: (1) highlight the important -- the relevant -- features in the descriptions, and (2) form general classes of episodes by collapsing together those which are similar.

Consider the highly schematized descriptions of two 'infield singles' depicted in Figure I.8.[4] For the sake of simplicity, only the final competitive interactions are shown. In Figure I.8a, we see that the batter was B6 and that this interaction took place at time 109 and 110. Figure I.8b portrays the same competitive interaction except that the batter is now a member of the A team, A1, and the time is 224 and 225. The system hypothesizes that the features 'goal', 'action', and 'relationship' are the important ones in the description of the events depicted in Figure 8. The basis for this inference stems from the following observation:

> The features which are relevant are those which will eventually be useful in hypothesizing goals and relationships for the players in a baseball game.

Feature values which are not identical and which are not hypothesized as relevant can be generalized. In Figure I.8, the values for the feature 'name-of-player' and the values for the feature 'exact-time-of-occurrence' differ in the two depicted episodes. The resultant generalized representation contains constrained variables which match any pair of players on opposing teams, and constrained variables which reflect the relative order of occurrence of the two acts (Figure I.8c).

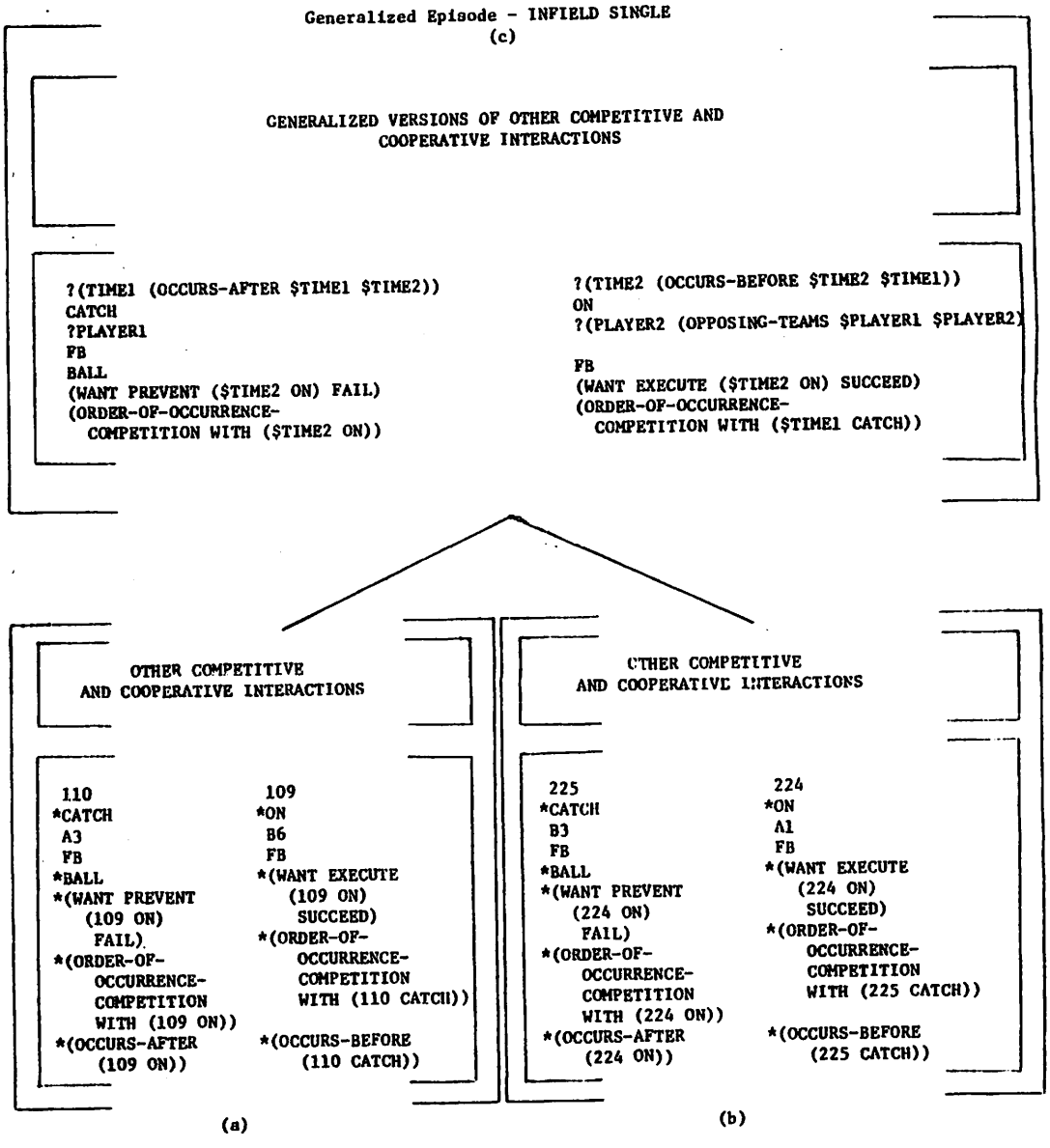


Figure I.8 Forming a General Class of Episodes

Episodes are represented by the competitive and cooperative interactions hypothesized to exist in the episode. In (a) and (b) above we include only the final competitive interaction for the episode 'infield single'. In (a) we see that a player, B6, reaches FIRSTBASE before an opposing player, A3, CATCHes the BALL. A class of infield singles is formed to reflect the similarities in the individual observed episodes; the similarities are based on specific features common to all episodes. In the generalized version of the final competitive interaction of an infield single (c) we see that the player feature has been generalized to a variable which matches any two players on opposing teams. The notation, which we will be using throughout this thesis, is interpreted as follows: variables, which match values, are indicated by the prefixes ? and $; constants, which match only identical constants, are not prefixed.

Our system does not employ an explicit teacher to partition the episodes into general classes. However, the system is provided with a definition for "similarity"; namely, two (or more) episodes are similar, and thus can be merged together to form a class, if they have identical values for the features hypothesized as relevant. Thus, the two instances of infield singles depicted in Figure I.8 can be merged together since they have identical values for the features 'goal', 'action', and 'relationship'. The episode classes discovered by the system after observing roughly a regulation, 9-inning game, are depicted in Figure I.9. A more detailed discussion of these results will be presented in Chapter VIII.

In general, we know that events (or objects) can appear to be dissimilar at one level of description, yet can appear to be similar at a more abstract level of description. A stool and a rocking chair have a different physical appearance; however, in terms of the coarsest description of the "function-of-the-object" (namely, to a person wishing to be seated) they are identical. Thus, by ignoring the differences in some of the physical features and matching only on the "function" features, a rocking chair and a stool can be grouped together into the same class. Our system employs this technique of feature selection to move to a more abstract level of classes; it forms an additional level of classes based only upon the final competitive goals in an episode. The choice of the final competitive goals in an episode, as opposed to the first or intermediate ones, reflects the observation that the final competitive goals

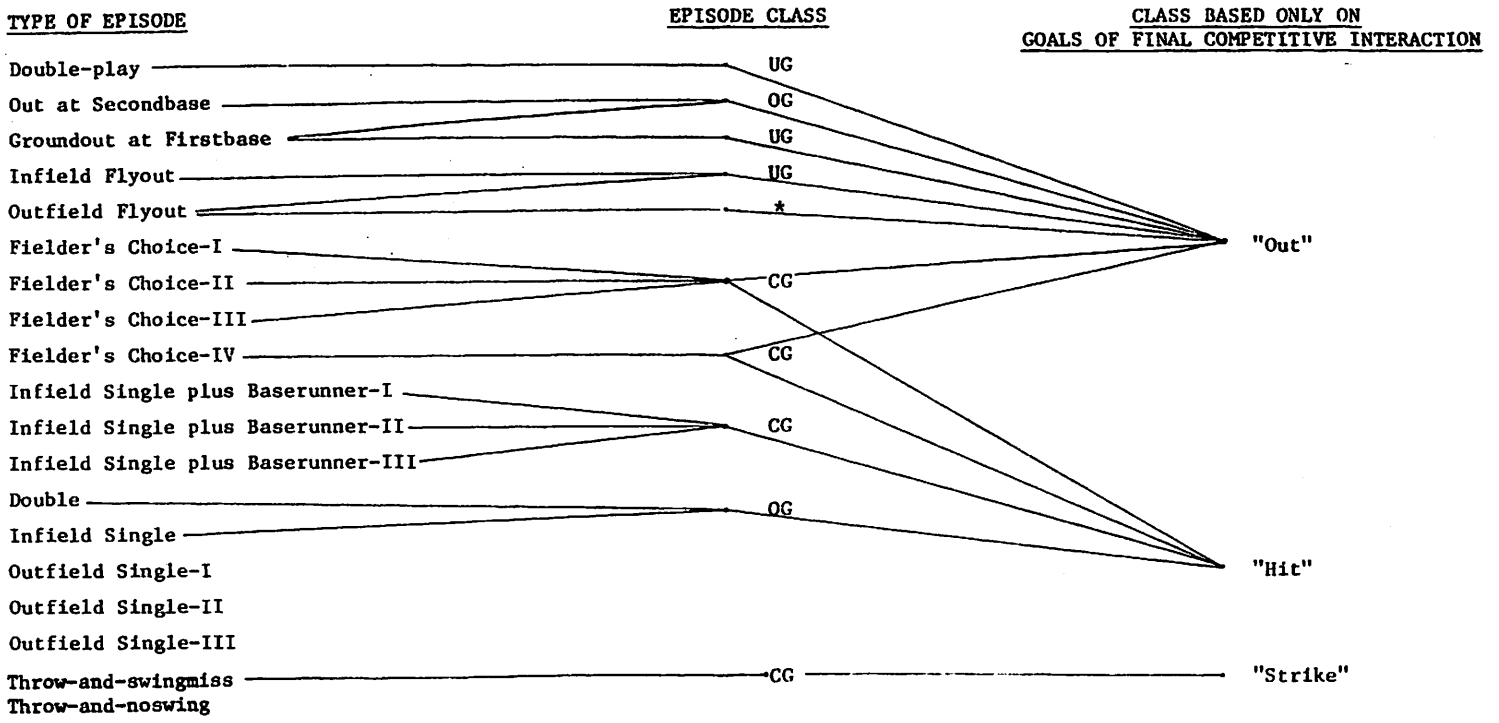


Figure I.9 Hierarchical Structure of Verified Classes

UG means undergeneralized, CG means correctly generalized, OG means overgeneralized, * indicates an episode class which should not have been verified since the analysis was incorrect.

summarize the episode. For example, the system creates a class of 'outs' by generalizing the classes of infield groundouts and outfield flyouts on the basis of similarity in the final competitive goal (Figure I.9). Note that this level of classification is based on a feature which was not initially observed, but rather, was hypothesized using the model of the domain.

I.4 The Model: Learning = Interpretation + Generalization

In this section we will try to step back from the specific example of the previous section and describe the model of learning used by the system. In so doing we hope to explicate the role of problem specific knowledge in the learning process. We shall begin this discussion with a quote from Jordan's [JOR68] commentary on A.N. Whitehead's view of explanation and generalization; it eloquently expresses the approach which we have taken to learning.[5]

> Faith in reason should not totter in the face of incoherence. Observers on Mars, without our knowledge, have planted a 'probe' with television cameras and are now watching a game of rugby football being played in England. They want some explanation of what the cameras are recording which will cohere with their general theories of what happens on our planet. The ranges of the cameras are not powerful enough for the Martians to see the ball; it appears that a lot of men in patterned shirts are performing a dance or orgy. The Martians' attention is drawn to the goalposts. They connect these with the similarly shaped objects to be seen on the roofs of some nearby houses.

> Now it is to be imagined that the Martians understand religious notions but have no sporting instincts. They conclude that the game is a religious dance rite and that the buildings with the H-shaped signs are temples.
>
> The roof signs, of course, are television aerials, and their resemblance to rugby goal-posts are accidental. The Martians are wildly mistaken. But their guess illustrates cohesion. They are trying to find meanings in the things which will lie together in a harmony that excludes the merely arbitrary. This is precisely the nature of the philosopher's faith in reason, a faith widely asserted in spited of the frustrations to which the above fantasy points.

One interprets, and thereby understands, new situations in the world in terms of the frame of reference that one brings to the learning situation. For example, our curious Martian friends used their understanding of religion and religious ceremonies to focus on specific features in the environments under observation; they attended to the T.V. antennas and the goalposts rather than the thousands of other features in the "country-side scene" and the "athletic-contest scene". They interpreted these features in the context of religion and then classified both scenes as similar, i.e., as different aspects of a religious ceremony. The knowledge which the Martians used to perceive the world permitted this classification -- the concept of religion was not "in" the observations, but rather, it was "in" the heads of the Martians.

The model of learning espoused in the above quote and implemented in our system is depicted in Figure I.10. Namely, learning is composed of two processes: interpretation and generalization. A general model

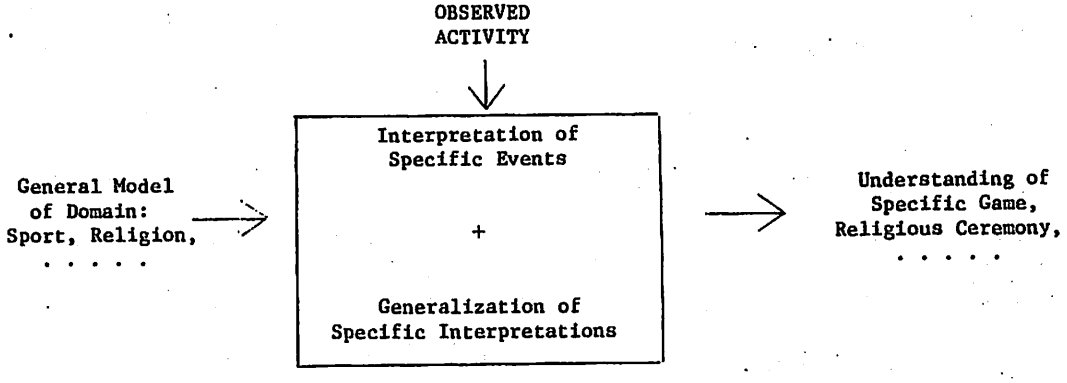


Figure I.10 Model of Learning Used by Our System

of a domain is used to construct an interpretation of the observed activity; terms from the domain model are added to the input descriptions. This interpretation, which is specific to the particular activity observed by the system, is then generalized by ignoring the non-crucial features. In this way, similar interpretations are merged together to form general classes of interpreted events.

The process of using stored knowledge to construct interpretations of observations can be viewed as the process of linking the "old with the new". That is, a system uses what it already knows in order to assimilate new information into the system. Once integrated, the new information can be used in the same ways as the original knowledge in the analysis of subsequent events. The mechanism for this linkage is a standard one: pattern recognition. Here we take this term in its broadest sense, matching an input description to a stored concept. Minsky [MIN63] has put it quite clearly,

> But by themselves, the simple learning systems are useful only in recurrent situations; they cannot cope with any significant novelty. Non-trivial performance is obtained only when learning systems are supplemented with classification or pattern-recognition methods of some inductive ability. For the variety of objects encountered in a nontrivial search is so enormous that we cannot depend on recurrence, and the mere accumulation of records of past experience can have only limited value. Pattern Recognition, by providing a heuristic connection which links the old to the new, can make learning broadly useful.

The stored concepts about competitive games can also be thought of as "expectations"; the model expresses what the system expects to see in a competitive game. The system uses that stored model in order to

recognize when a specific observation is an instance of one (or more) of its expectations. In the task of learning, those expectations are general rather than specific; the system does not initially expect to see a player try to HIT a BALL with a BAT, but rather it does expect to see two players on opposing sides in competition.

The generalization of specific descriptions consists of sifting through those descriptions to eliminate the irrelevant aspects and highlight the relevant ones. The use of general domain knowledge in that process reflects our view that the objective of a learning system is to discover the relevant features in the interpreted events -- not to discover just the recurrent ones. As we shall argue, rule induction systems which do not use domain knowledge -- or at least claim that they do not -- can only discover correlations between features present in the sample observations.

I.5 The Selection of Baseball as the Task Domain

An important consideration in the development of AI paradigms and techniques is the choice of task domain. Selection of a game domain such as baseball might appear to be frivolous. However, we feel that the action-oriented game of baseball has provided us with a rich set of behaviors in which to explore issues crucial to artificial intelligence (AI). It is a spatio-temporal world in which there is simultaneous and continuous activity. This activity involves human actors who have purposes and plans, and inanimate objects which obey physical laws. Thus, the Frame Problem [SRI76, McC69], the modeling of simultaneous and con-

tinuous activity [HEN73], and understanding causality [RIE73,SCHA73b] must all be considered. Moreover, whether human activity is perceived by reading stories or through actual visual observation, the same underlying processes must be performed -- namely, the inference of the goals and plans of the actors [SCHA75, SCHM76, RUM75] based upon their actions. Thus, the task domain of baseball games encompasses many of the important issues in the mainstream of AI research.

Regarding the complexity of baseball, concern has been voiced that it would be difficult for a human to acquire an understanding of the game on the basis of the information given to our system. On the contrary, we believe that many of the local goals of the observed actors often have only a few rather obvious alternative explanations. The complexity arises because it is difficult for the many local hypotheses to be integrated into a global structure by a human observer. Moreover, as the number of examples increases, so might the confusion. Nonetheless, we believe that, with patience, people can and would understand many of the important rudimentary aspects of this game -- much as our system does.

I.6 Important Issues in This Thesis

While the conclusions in Chapter IX need to be motivated by the intervening material, we would still like to let the "punch lines" out of the bag. We shall do this by posing a series of questions that will be addressed in the following chapters. The issues raised in these questions are the focal problems explored in this work.

(1) How can the process of learning be decomposed, and what can be said in general about such a decomposition? What roles do interpretation and generalization play in learning?

(2) How can old knowledge be used to assimilate new information? What representational problems arise in this process?

(3) How can new knowledge be used effectively by the system? What representational problems arise in the creation of active knowledge?

(4) How can a learning system funciton without an explicit teacher? How can a system construct a hierarchical classification scheme?

(5) How can hypotheses be verified? How can a system cope with interaction of generalization with verification? How can higher level knowledge be fed back to redirect the system's focus of attention?

Please keep these questions in mind. They will provide a general -- and welcome -- rallying point in the detail that will soon follow.

C H A P T E R I I

SYSTEM OVERVIEW

II.1 Multi-Level Architecture of the System

We have adopted what has come to be called a multi-level architecture [HAN76, LES77] for the computer implementation of the knowledge-directed learning strategy outlined in Chapter I. This structure was naturally suggested by the requirements of the task under investigation. Namely, in order to discover general rules describing the regularities in the relationships between players' actions and goals, the input pattern descriptions needed to be significantly transformed; while the descriptions of the observations were only in terms of perceptual features, the general rules must include high level features such as competitive and cooperative goals. Figure II.1(a) summarizes the various levels which the pattern descriptions passed through in the annotated example of Section I.3.

Clearly, the learning paradigm which we are exploring need not be implemented in a multi-level structure (e.g., [BUC77]). However, the flexibility of multi-level architecture allows -- in fact, encourages -- the molding of the system to the problem, and not the other way around. In decomposing the system into multiple levels of (1) pattern description, (2) knowledge, and (3) processing, we have been able to tailor each level to match the requirements of the logical subproblems.

In the following sections we shall attempt to provide a brief description of the various levels in the system. The following chapters will expand on the arguments made here. First, the role which levels of pattern descriptions play in our system is discussed. Next, an extensive discussion of the knowledge and processing levels is presented.

II.2 Levels of Pattern Descriptions

A pattern description is simply a set of features (Figure II.1(a)). The multiple levels of pattern descriptions perform two roles which are needed in any complex system:

(1) they act as a medium of communication between processing levels, and

(2) they define the input/output characteristics of the various processes.

For example, the pattern descriptions of snapshots input at Level 1 are transformed into filtered snapshots at Level 2 via the application of the heuristic "change is interesting" (Figure II.1(c)). Similarly, successive levels of processing transform the pattern descriptions in other appropriate ways.

II.3 Levels of Knowledge and Processing

II.3.1 Attention Focussing

As its name implies, the purpose of Attention Focussing (Figure II.1 (d)) is to focus processing on "interesting" aspects of the observations. From Figure II.1(b) Level 1, we see that the input to this stage of pro-

cessing is the "raw" observations while the output is those aspects of the observations deemed interesting.

In moving from Level 1 to Level 2 Attention Focussing (AF) attempts to reduce the data by filtering out actions on the basis of the following heuristic: change is interesting. Animals too seem to employ this heuristic, and thus they tend to habituate to those features of the environment which are unchanging. Similarly, our system filters out the actions which do not change from snapshot to snapshot, except those which require a significant amount of skill and/or energy to perform, e.g., RUNning FAST. The results of this filtering algorithm are quite dramatic. The number of pattern descriptions per snapshot is decreased from 24 to an average of 2 or 3. Certainly, low energy action sequences that do not change can be important. On a first pass, however, the system will miss such subtleties. The hope is that later processing will redirect the system's attention to take note of non-changing activity when necessary (see Section V.3).

Next, AF attempts to segment the continuous sequence of snapshots into units (Level 3) which are meaningful in the task domain. Such units are called episodes and are carved from the snapshot sequence on the basis of the following heuristic: a competitive episode is often indicated by a period of high energy activity surrounded by periods of low energy activity. In otherwords, the cycles of competitive activity in a game can often be distinguished from the ritualistic or preparatory activity by the difference in the degree of energy expended. An 'infield single' or a 'flyout' would be typical episodes in baseball; a 'down' would be an

episode in football. The episode partitioning is crude and later stages provide more semantic analysis in order to punctuate the boundaries more clearly.

In Figure II.1(d) we also see a bidirectional arc indicating flow of information from Hypothesis Evaluation (Predictions) to Attention Focussing, and a flow of information (Results) in the opposite direction. Predictions of events serve to redirect the system's focus of attention. Results from these predictions are used by Hypothesis Evaluation in the verification of the hypotheses on which the predictions are based.

Eventually, when the system has verified its hypothesized understanding of various aspects of the observations, this stage of processing will use those verified rules as templates to recognize specific action/goal sequences. For example, instead of seeing pitcher THROW, batter HIT, batter RUN, etc., the system will see 'infield single' or 'infield ground-out'. That is, the lower level features will be built up hierarchically into composite features. While the system will clearly have to observe the lower level features, the point is that it will be able to label specific sequences of them with higher level conceptual names.[6]

II.3.2 Hypothesis Generation

During this next phase of processing, the system attempts to provide explanations for the observed events in terms of the general knowledge with which it is initially supplied. As we saw in Section II.3.1, the initial interpretation performed during Attention Focussing resulted in the elimination of some pattern descriptions from the observations.

During Hypothesis Generation, however, interpretation results in the addition of two types of features to the remaining pattern descriptions. The first type deals with the inference of simple causal relationships between actions (Level 4) while the second type deals with the hypothesis of competitive and cooperative goals and relationships (Levels 5 and 6).

II.3.2.1 Inference of Non-purposive Relationships

A common-sense understanding of the "physics of actions" is provided to the system; this knowledge defines the cause and effect relationships that can link actions together (Level 4). In particular, the system is supplied with an act-schema for each observable action. The procedural component of an act-schema defines the particular preconditions and consequences pertaining to that action. For example, in the action sequence

[(102 THROW A1 PM BALL)(SWINGHIT B1 HP BALL)].

The act-schema for the action SWINGHIT can discover that the preceding THROW set up the pre-conditions for the execution of the SWINGHIT. The recognition of such a causal enablement relationship is reflected in the addition of a new feature to the pattern descriptions of the participating actions (Figure II.1(a), Level 4). An act-schema also has declarative information which can be "read" by other procedures. Later it will be important to know how difficult it was to execute a particular action; information relevant for such decisions is included in the act-schemas.[7] Finally, note that the information contained in the act-schemas is independent of competitive games; the act-schemas just describe the relationships between actions in a spatio-temporal world.

II.3.2.2 Inference of Competitive and Cooperative Goals

We have developed a first-order theory of competitive action games. While this theory is relatively simple, it does describe the basic knowledge which is needed to understand the goals and relationships in a competitive action game (Figure II.1(c), Level 5). The following are examples of facts in that theory:

i. Participants in this type of activity engage in 'competitive' and/or 'cooperative' relationships.

ii. In a 'competitive relationship', players on one team try to prevent their opponents from achieving their goals.

iii. In a 'cooperative relationship', players on one team try to help each other succeed in their respective goals.

It has been possible to define a small number of competitive and cooperative relationships, together with the goals indicated by such interactions. We do not contend that this set covers all relationships in all games; such a feat would be a major contribution to the study of games (cf., [AVE71a]). On the other hand, the defined set is not limited to baseball. We shall endeavor to give examples of interactions from other games which can be interpreted in terms of the relationships which have been defined.

The following rule illustrates the type of general knowledge about interactions supplied to the system. It was employed to hypothesize a competitive relationship between the pitcher A1 and the batter B1 in the annotated example of Section I.3.

IF (i) a player on one team physically enabled a player on an opposing team to execute an action, and if

(ii) both actions are considered to be difficult to perform (i.e., require a measure of skill and/or energy), and if

(iii) the player, who enabled the observed action, had performed his difficult action with a greater degree of skill and/or energy, it would have decreased the likelihood of his opponent's being able to execute the observed action,

THEN HYPOTHESIZE THAT

(i) the two players are in a competitive causal relationship,

(ii) the goal of the player who enabled the opponent's action was to prevent the opponent's action, thus this player failed with respect to achieving his goal in this instance, and

(iii) the goal of the opposing player was to execute the observed action, thus this player succeeded with respect to his goal in this instance.

This rule does not pertain only to baseball; the competition involved in volleying in a tennis match would also be discovered by this rule. Also note that the rule does not specify tests for specific actions, but rather tests for properties of actions. Rules of this sort are called Causal-Link Schemas (CLSs), and are implemented as production rules[8] [NEW73].

If the properties of an action sequence satisfy the conditions in a CLS (or CLSs), the result would be a further augmentation of the participating pattern descriptions; new features such as goal and causal relationship would be added to the descriptions (see Figure II.1(a), Level 5). However, one of the main objectives of our learning system -- and of

learning systems in general -- is the future use of acquired knowledge. This implies that the passive pattern descriptions produced by a CLS must be turned into active procedures. This is accomplished by having a CLS produce a version of the augmented pattern descriptions in the form of a production rule. Thus a hypothesis of a competitive or cooperation interaction by a CLS results in the creation of two representations which capture the same information. The system employs both representations since each is best suited to a particular role; forming general classes is more easily done in terms of sets of features -- pattern descriptions, while actively using knowledge requires procedures. Since the number of such hypotheses is not great, the system is in no danger of being unduly burdened.

It is not by accident that both the initial general rules (Causal-Link Schemas) and the hypothesized specific rules have the same structural form: production rules. The homogeneity in representation of both new and old knowledge facilitates the application of acquired information. Without modification, the interpreter which applied the old knowledge can apply the new knowledge.

Our learning system derives a unique benefit from the levels architecture with respect to the problem of knowing when to use acquired knowledge. In this organization all knowledge at a level serves the same general function. In particular, all the CLSs hypothesize goals and relationships (Level 5, Figure II.1(c)). Similarly, the hypothesized rules themselves also make hypotheses of specific goals and specific relationships. Thus

the new information can be integrated into the same level as the old knowledge, and used whenever the old knowledge is to be applied.

Episodes have internal structure; the various competitive and cooperative goals are achieved in order to accomplish the final competitive goal. The final competitive goals in an episode summarize the activity of an episode. This additional structure is highlighted in our system; a separate level is created in which only the final competitive goals of an episode are represented. This level will be used during Hypothesis Generalization to create classes based only on final competitive goals of episodes (see Figure II.1, Level 6).

The strategy for discovery employed by our system is different in character than the heuristic search paradigm used by some AI discovery systems (e.g., [BUC73, LEN76]). We do not employ an algorithmic generator of hypotheses (e.g., the DENDRAL algorithm in the Meta-DENDRAL System [BUC73]), with knowledge of the domain brought in to prune the search tree. Rather, we use knowledge of the domain directly to assure the generation of plausible hypotheses. Since alternatives are produced, some search for the correct one is necessary. However, the nature of that search is not one of pruning a search tree, but rather of evaluating several plausible hypotheses. In our approach, then, the prior knowledge of the system constrains what can be discovered -- its expectations direct discovery.

II.3.3 Hypothesis Generalization

The input for Hypothesis Generalization are the specific pattern descriptions output from Hypothesis Generation. The induction phase serves to transform this input into general rules by highlighting a distinguished subset of features in each description. In so doing, interactions and episodes which are similar with respect to the distinguished features can be merged together into general classes.

Two strategies for generalization can be identified: data-directed generalization (DDG) and knowledge-directed generalization (KDG). In the former, a general rule is created by matching a set of exemplars (e.g., pattern descriptions of events) together, and abstracting the features common to all the exemplars. This technique can be computationally costly, since the number of matches required and the number of alternative generalizations produced can become quite large. Data-directed generalization is a bottom-up process; the decision to include a feature in the general rule depends only on the presence or absence of that feature in the data. Thus, rules formed in this way consist only of features which repeatedly co-occur. However, as the quote from Minsky [MIN63] in Chapter I suggests, recurrence may not be enough; additional factors may be required in order to highlight the relevant features as opposed to those which simply recur. In Chapter VI, we shall return to these issues and discuss them in greater depth.

The knowledge-directed approach to generalization, we claim, serves to remedy the problems and limitations of the data-directed approach. In

this method, knowledge of the task domain is explicitly used to direct and constrain the matching process, and select the relevant features comprising the best generalization. Clearly, knowledge must be used carefully, lest the learning process be compromised. In Chapter VI, we shall examine several systems [WAT70, WIN70, EVA68, FIK72, SUS73] which use KDG.

In our system, the general knowledge initially provided to the system about competitive action games is also used to highlight the relevant features in the data (pattern descriptions). The features which are relevant in our task are those which play a role in determining the goals and relationships of the players in the game. Thus, the color of a player's eyes would most likely not be relevant, while the degree of difficulty of a physical action most likely would. The selection of the relevant features is done by the Causal-Link Schemas during Hypothesis Generation. Recall that a CLS hypothesizes goals and a relationship. In order to determine if such hypotheses should be put forth in a given situation, the CLS tests various aspects of that situation. Those features to which the CLS refer are, by definition, relevant to the task of inferring goals and relationships. Thus, the features in the resultant pattern description which are hypothesized to be relevant are just those which are addressed by the CLS.

Figure II.2 depicts the final competitive interactions hypothesized to be present in two instances of the episode 'infield single'. The CLS, which knows that the order in which two actions occur may be important, has made the following hypothesis (Figure II.2a): the reason that B6 was

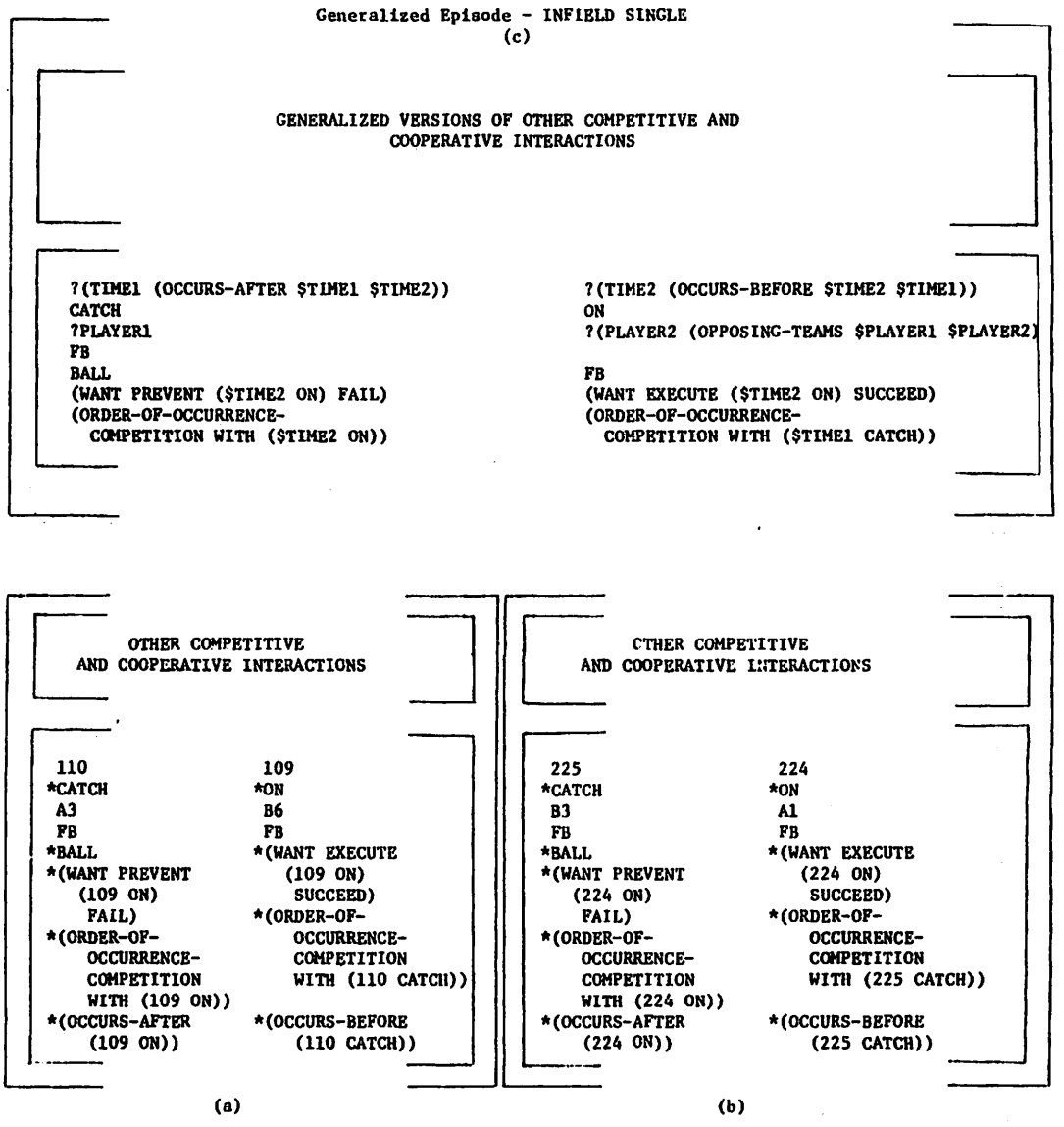


Figure II.2 Forming a General Class of Episodes

In (a) and (b) above we include only the final competitive interaction for the episode 'infield single'. In (a) we see that a player, B6, reaches FIRSTBASE before an opposing player, A3, CATCHes the BALL. A class of infield singles is formed to reflect the similarities in the individual observed episodes; the similarities are based on specific features common to all episodes. In the generalized version of the final competitive interaction of an infield single (c) we see that the player features has been generalized to a variable which matches any two players on opposing teams. The notation, which we will be using throughout this thesis, is interpreted as follows: variables, which match values, are indicated by prefixes ? and $; constants, which match only identical constants, are not prefixed.

allowed to execute ON FIRSTBASE was due to his performing that act before A3 caught the BALL. Thus, the features 'occurs-before', 'occurs-after', 'goal', and 'action' are hypothesized as relevant (indicated by asterisks), while the features 'location' and 'name-of-player' are not. Those features which are not hypothesized as relevant can be generalized; constants can be replaced by variables in order to accomodate the differences in the data. In Figure II.2c, we see that the features 'name-of-player' were replaced by variables which match any pair of opposing players.

The episodes observed by our system are not partitioned into classes by a trainer. Rather, the system itself forms the episode classes by merging together those episodes which have the same values for those features which are hypothesized as relevant. The different sets of relevant features serve to partition the episodes into classes meaningful to the task domain. In Figure II.3, the final competitive interactions of an 'infield single' and an 'infield groundout' are depicted. These two episodes are similar with respect to some competitive and cooperative interactions. Nonetheless they cannot be merged together since their final competitive interactions have different values for the features 'occurs-before', 'occurs-after', and 'goal'.

We know from our everyday experiences that events which appear different at one level of description can appear the same at another, more abstract level of description. In the context of baseball, an 'infield groundout' appears to be different from an 'outfield flyout' at the episode level of description (Level 7). However, under the premise that the

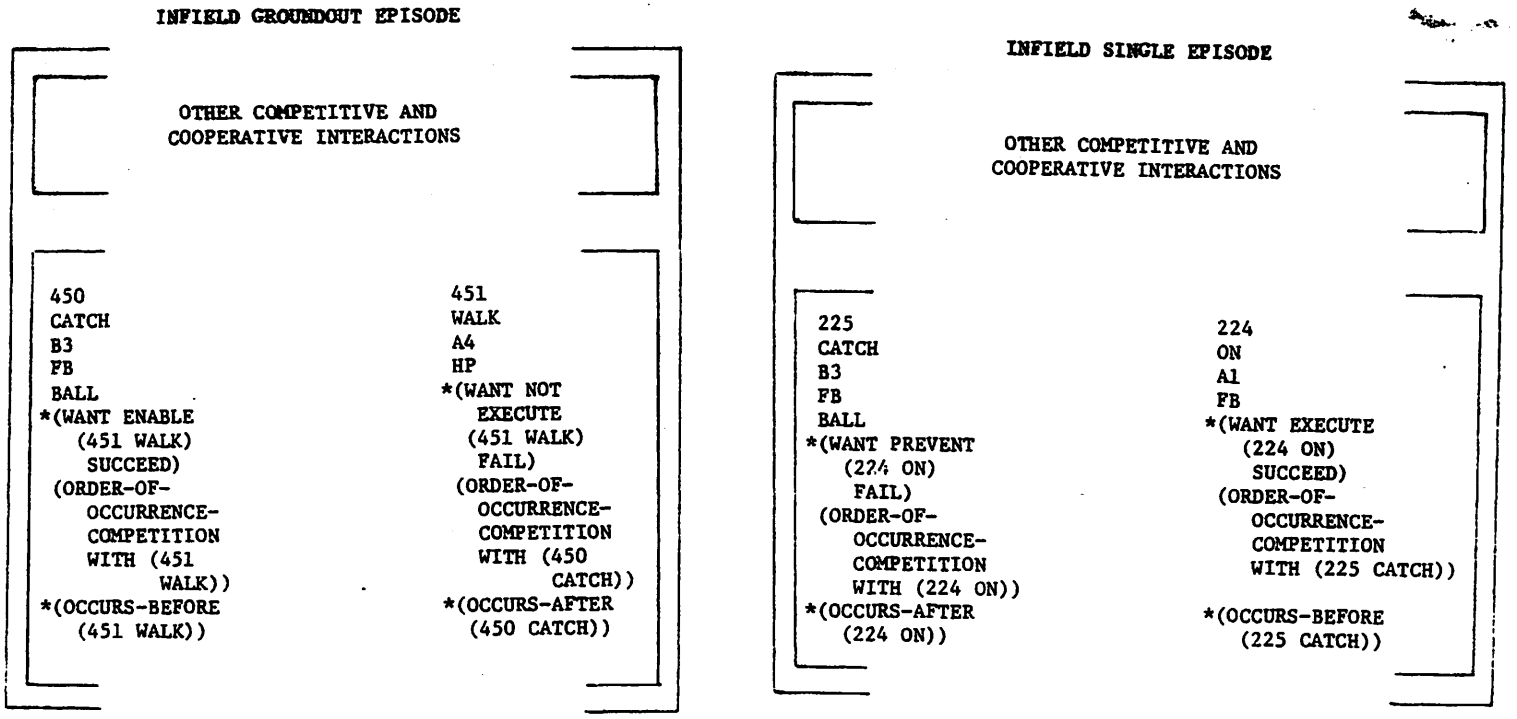


Figure II.3 Dissimilar Episodes Cannot Be Merged into a Single Class

Those episodes which match on the features hypothesized as relevant can be merged together. In the above figure, however, the final competitive interaction of infield groundout episode differs from the corresponding competitive interaction in the infield single-episode at key points: relative-time of occurrence and goal. Thus, they cannot be merged together.

final competitive goals of an episode summarize the activity in the episode, we can form a more abstract level of description based only on this descriptive feature. At this level of description (Level 8), then, an 'infield groundout' and an 'outfield flyout' can be merged together into the same class, since they both have the same final competitive goals, i.e., while the batter wants to reach FIRSTBASE, the players in the field want to prevent him from doing so.

II.3.4 Hypothesis Evaluation

Hypotheses are just that -- unverified conjectures which quite possibly are wrong. Indeed, there are often conflicting interpretations for the same events. Thus hypotheses must be verified before they can be accepted as truth. The basis for this decision is the confidence value of the hypothesis. This measure of the evidence supporting a hypothesis is a simple function of two factors: results from the predictions generated from a hypothesis, and consistency of a hypothesis with previously acquired (verified) knowledge. An externally supplied threshold is used to select truths; hypotheses whose confidence values are above threshold are considered to be correct. Chapter VIII illustrates the system's performance under varying threshold settings.

Predictions of events and their interpretations are made based on the hypotheses generated earlier. All predictions are implemented as production rules and are fed back to Attention Focussing for matching against incoming data. Results of the predictions are used to modify the confidence values of the hypotheses appropriately.

Two types of predictions are made. One type predicts the complementary success/failure outcome of a competitive interaction. For example, when the batter HIT the BALL, the system hypothesized that the pitcher A1 FAILed and the batter B1 SUCCEEDed. Based on the definition of competition, a prediction can be drawn from this hypothesis. Namely, if this hypothesis is correct, then the system should observe an interaction in which the pitcher THROWS the BALL and the batter does not HIT the BALL, and the interpretation hypothesized for this event is that the pitcher SUCCEEDed with his goal while the batter FAILed. Successful matching of these predictions serve to increase the confidence value of hypotheses.

If a hypothesis is correct, then certain events should <u>not</u> happen. The second type of predictions which are produced are based on the logical properties of a hypothesis. If any of these predictions are found to match in the subsequently observed data, then the confidence value of the hypothesis from which that prediction was generated, is made to go negative. One piece of falsifying data serves to eliminate a hypothesis; confirming instances only serve to increase the confidence values.

In addition to prediction, internal consistency among hypotheses affects the confidence values of the hypotheses. A hypothesis which is accepted as "truth" is used to increase the confidence values of those hypotheses which are consistent with the new truth, and decrease the confidence values of those inconsistent with the new truth. For example, when the system decides that getting ON FIRSTBASE is a desirable goal (for one team), then the confidence values of those hypotheses which suggest that getting ON FIRSTBASE is undesirable are decreased.

While the evaluation of hypotheses is itself a difficult task, it is further complicated by the introduction of generalization. That is, the predictions reflect the generalizations introduced into the hypotheses. If the generalization is incorrect, then the predictions may or may not be tuned to the appropriate data. Since straight interpretation systems do not perform generalization, and since learning systems of which we are aware assume the correctness of their data, this problem has not been explored in the literature. In subsequent chapters (especially Chapter VIII) we shall examine this issue more closely.

CHAPTER III

THE COMMON-SENSE PHYSICS OF ACTIONS: CONTENT AND REPRESENTATION

Before the system can plunge into an analysis of the observed activity in terms of its model of competitive games, it must first be able to understand the basic relationships between actions in a spatio-temporal world, e.g., the effect of a person throwing a ball is to cause that object to travel some distance on the ground and/or in the air; it is more difficult to grasp and hold onto a ball moving at a high velocity, than a ball moving at a low velocity. Facts such as the above are provided to the system initially; they constitute, what we call, the common-sense physics of actions. A child learning to deal with the world would develop a similar common-sense grasp of the relationships between objects, persons, and actions.

It is important to note that this type of knowledge about actions is independent of and prior to competitive action games. However, action games like baseball play heavily on such facts. For example, it is a fact that hitting a small spherical object with a stick-like object for a distance greater than 400 or so feet in the air is a difficult task for a human. Thus in baseball, a person accomplishing this feat is highly rewarded; namely, a "score" is automatically afforded to the player. Similarly, a model of religion could also provide a reasonable interpretation for such phenomena. In the above example, one could say the following: since it is a fact that hitting a ball a distance greater

than 400 feet is difficult, men who can perform such supreme acts must be members of the priesthood or in some way be endowed with mystical powers. In the world of professional (commercial) athletics one sometimes wonders which model of interpretation should actually be employed.

The contents of this chapter are as follows: a description of the system's representation of an observed baseball game will be presented in the first section. Next, a description of the content and representation of the knowledge about the observable actions (e.g., RUN, CATCH, THROW) will be presented. We conclude with some observations drawn from this chapter.

III.1 Representation of the Spatio-Temporal Environment: The Game of Baseball

Input to the system is supplied by a program that simulates the continuous game of baseball by breaking it up into discrete time intervals, called snapshots. Each snapshot consists of a set of pattern descriptions depicting the state of the world at that instant in time. A pattern description is a 5-tuple which captures 4 essential perceptual dimensions (features) of such a miniworld -- action, actor, location, time-of-occurrence, plus any modifiers to those dimensions.

Table III.1 lists the set of actions observable by our system. They are the natural ones for describing an action-oriented game, e.g., RUN, THROW, CATCH, etc. Descriptors like INNING, RUNS, etc., represent the various markers on the scoreboard. Note that the machine does not initially understand the semantics of such markers. Similarly, the system

```
(time HOLDOBJECT player location object)
(  "  THROW       "       "       "  )
(  "  SWINGHIT    "       "       "  )
(  "  CATCH       "       "       "  )
(  "  SWINGMISS   "       "       "  )
(  "  WALK        "       "    modifiers)
(  "  RUN         "       "       "     )
(  "  ON          "       "       "     )
(  "  AT          "       "       "     )
(  "  AIRMOVING  object    "    modifiers)
(  "  GROUND-
       MOVING     "       "       "     )
(INNING number)
(RUNS-TEAMA number)
(RUNS-TEAMB   "   )
(OUTS         "   )
(STRIKES      "   )
(BALLS        "   )
(HITS-TEAMA   "   )
(HITS-TEAMB   "   )
```

Table III.1 List of Observable Actions and Scoreboard Descriptors

Events are described in terms of four basic properties: action, actor, location, and time-of-occurrence (from a fixed starting point). There is also a fifth aspect, modifiers, which can accompany a description of an event, e.g., speed of object. In total, these features compose a pattern description of an event. Symbolic names in lower case letters indicate variables, while symbolic names in upper case letters indicate constants.

represents locations such as pitcher's mound, first base, etc., only as X-Y coordinates; they have no a priori significance. The value of the feature 'time-of-occurrence' is a number which encodes the time the event occurred in the game, e.g., the first event has time 1, the second event time 2, etc. As we shall see, the system will also be able to describe the time an event occurred in terms of a relation to the time another event occurred. If the object (player or ball) is continuously moving, the 'location' feature encodes the object's initial position plus directional information; when necessary the system can calculate the object's actual position.

Figure III.1 illustrates 3 sample snapshots. There are about 2000 snapshots in a "typical" game of baseball, with 24^9 pattern descriptions in each snapshot. In snapshot 102 we see player A1 THROWing a BALL. That same BALL is MOVING through the AIR in snapshot 103, and an opposing player, B1, is seen SWINGHITting the BALL in snapshot 104. The system knows initially that there are two teams of players (the A team and the B team) involved in the activity.

Questions can legitimately be raised at this point concerning the initially provided description of the world. We shall raise and attempt to answer the most serious ones. In general, however, we do not feel that our exclusion of some features or our choice of level of description seriously detracts from the model of learning which we are developing.

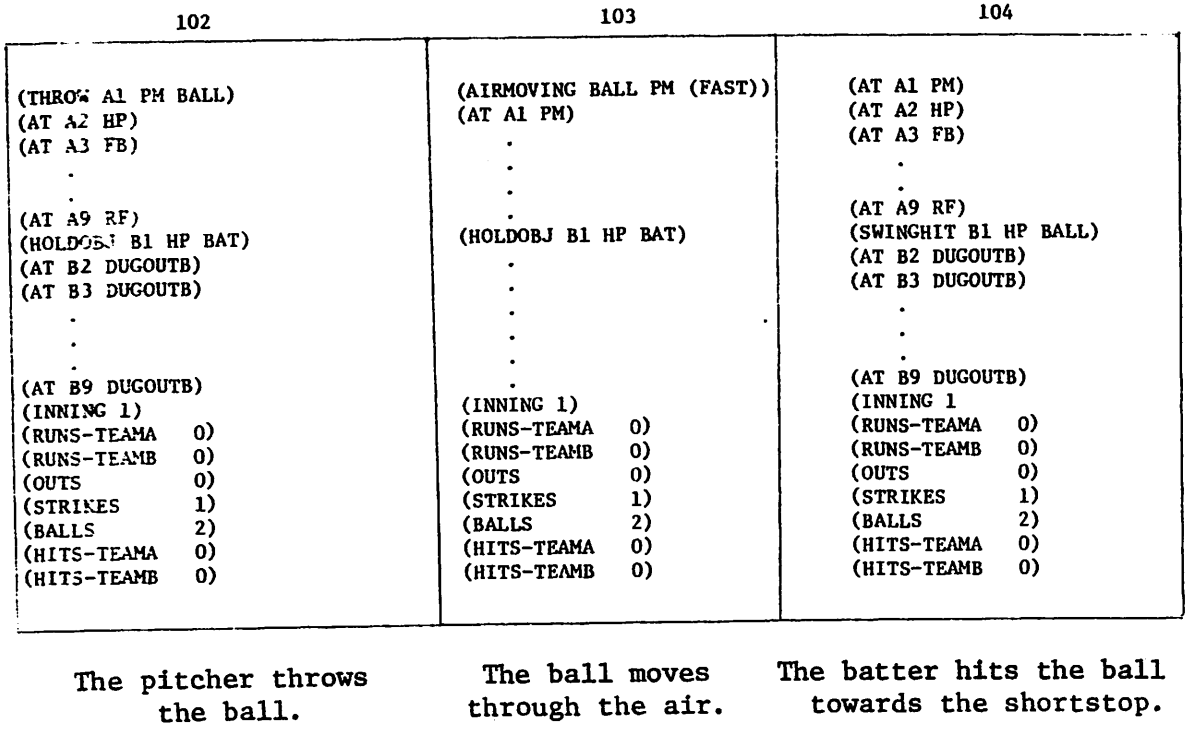


Figure III.1 Sample Unfiltered Snapshots

Each snapshot describes the activity of each player at a moment in time. Symbolic names for locations such as PM (pitcher's mound) are used for the reader's convenience; the system knows them only as X-Y locations. HP stands for homeplate, FB for firstbase, RF for right field.

Question: Why choose this particular set of perceptual features? For example, why not include the BEER-MAN-HAWKING-BREW or CLOUDS-MOVING? Why choose those 4 dimensions of the world to include in a pattern description? Why not encode the height, weight, hair color, etc. of the various team members? Isn't this over-simplifying the task?

Answer: We know from our own experience that we often perceive new situations in an "appropriate" way. If we did not, then we would have great difficulty in understanding the new situation. We bring a wealth of experience and knowledge to filter out the activity that probably is not relevant to our particular goals. Our system is initially tuned to activities on the field and the proper numeric markers. We believe most people roughly approach the game with this focus. The "beer man" is probably associated with a buy-sell setting, while "clouds moving" is probably associated with a natural outdoors setting. Thus, in an action-oriented game setting, we probably would not initially perceive such features as being relevant. However, when attempts at comprehension of a situation fail, then we might try to incorporate the features passed over initially.

Even though we have limited the number of features, the combinations of the existing features are still quite great. The interesting question of whether the system can discover the relationships between the relevant features still remains.

Question: In real life, players do not simply stand rigidly AT some location. Rather, they move about somewhat. Why do you ignore these perturbations?

Answer: From the sample snapshots in this chapter, it is clear that we have applied a preprocessing filter to the actions of the players. For the most part, such perturbations can be considered to be random motion; an outfielder pacing or a pitcher scratching his head are stylistic actions rather than competitive actions. If we had input this lower level of description to the system,

we would have had to include explicit knowledge that would have hypothesized that such perturbations are essentially irrelevant. While this touch may have been nice, it would seem to only complicate our task. In any case, our model of learning would have remained the same.

Question: Why this particular level of description? Why not describe the actions in terms of arm movements, muscle fiber actions, etc.? (This question is similar to the above question.)

Answer: Were we to perceive baseball in terms of limb movements and not in the "higher" level units such as walk, run, etc., we would have more difficulty in learning to understand the game. This is not to say that we would not or could not build the structure starting at this level. Rather, it would probably take longer and would definitely require an understanding of the relationships of "micro-actions" such as arm movements to "macro-actions" such as throwing.

One could make the argument that a physical object should be described in terms of atoms, molecules, and the like. However, a description of the world must start at some level; the interesting question is the relative level of description between concepts formed and the features initially observed.

The game of baseball which our system observes is a simplified version of the real game. Table III.2 lists the events which the system actually observes while Table III.3 lists some of the events which the system does not observe. The major reason some events were not included was due to the lack of knowledge needed to interpret them as events in a competitive action game. The deficiency in the knowledge base stems in part from our desire to simplify the problem, and in part from our

Episode Type
Infield Single
Infield Groundout
Outfield Single
Infield Flyout
Outfield Flyout
Outfield Double
Out at Second Base
Infield Single plus One Baserunner
Double-play
Fielder's Choice - Safe at Firstbase Out at Secondbase
Fielder's Choice - Out at Firstbase Safe at Secondbase
Pitcher Throws - Batter Swings and Misses
Pitcher Throws - Batter Does Not Swing

Table III.2

Episode's Observed by the System

Episode Type
Homerun
Triple
Foul Ball
A Hit with More Than One Baserunner
Sacrifice Flyout
Infield Ground Rule

Table III.3

Episode's Not Observed by the System

ignorance of how to specify in a general way the knowledge necessary to understand some events.

III.2 Understanding Actions and Their Relationships

III.2.1 Primitive Actions

If one looks at the consequences (effects) of actions, one notices that the consequences of ostensibly different actions are actually the same. For example, the consequences of the actions THROW and SWINGHIT are the same; first, the object propelled (a ball in our case) is MOVING, and then it comes to rest (or its movement is halted) at some location. Following this observation, we have represented the observable actions, such as THROW, SWINGHIT, in terms of the primitive actions which underlie them. Figure III.2 depicts the hierarchical structure which results from classifying actions on the basis of similarity of consequences.

Similar observations about the apparent commonality of underlying meanings have led some researchers to develop theories of understanding based on conceptual primitives (e.g., [SCHA73a]). We do not want to become embroiled in the controversy over the "reality" of primitives (see [WIL75] for a discussion of this issue). Rather, we simply contend that their use can aid processing in a system. In particular, our system derives the following benefits from such a natural organization:

(1) The specification of the relationships of an observable action need not mention any other specific observable action, since such a definition can be expressed in terms of primitive actions. Thus, observable actions can be defined relatively independently of any other

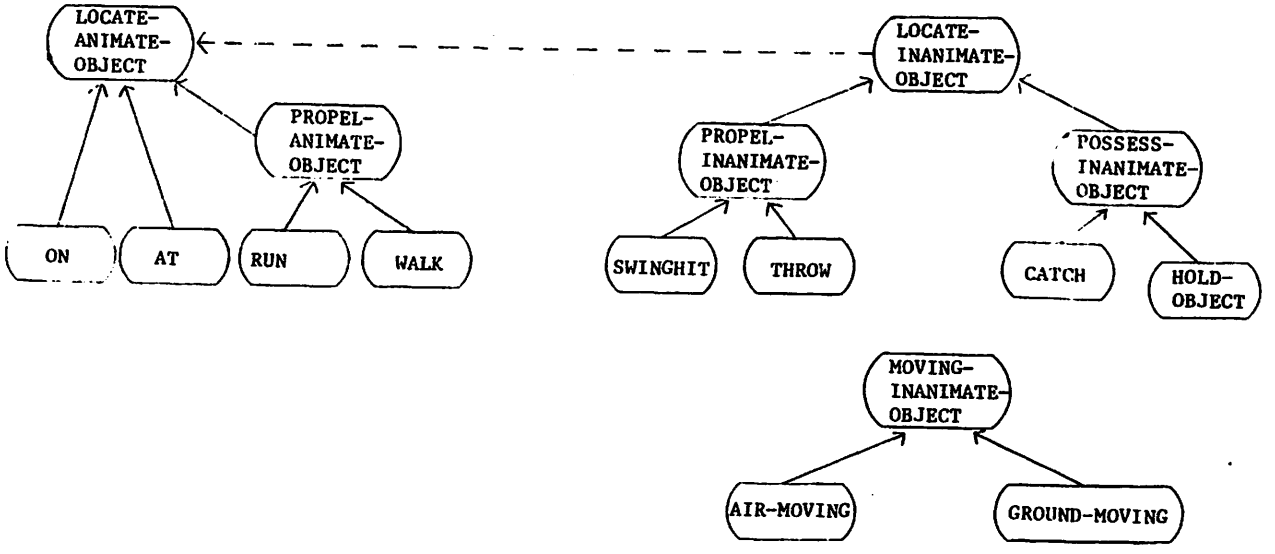


Figure III.2 Hierarchical Structure of Actions

The observable actions can be grouped into classes which reflect their common primary outcome; THROW and SWINGHIT are both in the class PROPEL-INANIMATE-OBJECT since the primary consequence of both actions is to cause an object to move. The semantic-net representation is interpreted as follows: the solid arcs connecting nodes are ISA-INSTANCE links, while the dotted arc is a COULD-COUNT-AS-AN-INSTANCE link. The latter arc indicates that some actions can have secondary consequences; SWINGHIT and THROW can be seen as fixing the location of the actor, hence they could be members of the class LOCATE-ANIMATE-OBJECT.

observable action. For example, we did not want to specifically list SWINGHIT as an action which could occur after the action THROW. This seemed both unnecessarily restrictive and unfair; the knowledge base would then be too specific to action sequences in baseball. To be more general, all other possible actions which could serve as events following a THROW, would need to be listed -- clearly, an unsatisfactory solution. Use of primitives avoids this nasty problem.

(2) The knowledge base is also relatively independent of particular action sequences in baseball. Definitions of other observable actions could easily be integrated into the system to enable the observation and interpretation of other games. We also do not claim any special status -- such as uniqueness -- for the set of primitives or their particular organization; they serve our purposes and do not seem unreasonable.

Consider again Figure III.2. One can interpret this description in terms of nodes and arcs of a semantic net [HEN76, WOO75]. The values of nodes can be either observable actions or primitive action types. Both types of arcs indicate membership in a class, e.g., the observable action THROW ISA-INSTANCE-OF the class of actions of the type PROPEL-INANIMATE-OBJECTS. The arcs are also transitive. Thus, THROW is a member of the class of actions of type LOCATE-INANIMATE-OBJECT by virtue of being a member of the class of PROPEL-INANIMATE-OBJECT actions.

The COULD-COUNT-AS-AN-INSTANCE-OF arc connecting the class of actions LOCATE-INANIMATE-OBJECT to the class of actions LOCATE-ANIMATE-OBJECT (Figure III.2) requires that a distinction be made between the primary and secondary consequences of an action. The former express the major function of the action, while the latter "clean up the details". For

example, the primary consequences of actions such as CATCH and SWINGHIT refer to the actors' interactions with an object; since they both serve to fix the location of the object they are members of the class LOCATE-INANIMATE-OBJECT. However, as a secondary consequence (side-effect) such actions also encode information about the location of the actor; thus, they COULD-COUNT-AS-INSTANCE-OF LOCATE-ANIMATE-OBJECT. As we shall see (Section IV.3.3), our system will need to make explicit use of this distinction.

III.2.2 Act-Schemas

An act-schema for each observable action is initially supplied to the system. An act-schema specifies four types of information: (1) the primitive action class to which it belongs, (2) the primary and secondary enabling conditions (preconditions) for execution of that action, (3) the primary and secondary consequences of executing that action, and (4) additional descriptive information about the action; e.g., the skill and energy required to perform the action, the range of alternative consequences, etc. The first three types of information are implemented as procedures which are evaluated for their effects, while the fourth type is declarative information which is accessed by other procedures.

Consider the act-schema for THROW in Figure III.3. There we see under the indicator PRIMITIVE-ACTION-TYPE that a THROW ISA-INSTANCE-OF the primitive action PROPEL-INANIMATE-OBJECT and that a THROW COULD-COUNT-AS-AN-INSTANCE-OF LOCATE-ANIMATE-OBJECT. Notice that the enabling conditions and the consequences are phrased in terms of primitive actions

```
THROW

  PRIMITIVE-ACTION-TYPE

    (ISA-INSTANCE-OF PROPEL-INANIMATE-OBJECT)
    (COULD-COUNT-AS-A-INSTANCE-OF LOCATE-ANIMATE-OBJECT)

  ACTION

    (?TIME2 THROW ?PLAYER2 ?LOCATION2 ?OBJECT2)

  PRIMARY-ENABLING-CONDITION

   (?(TIME1 (JUST-BEFORE $TIME1 $TIME2))
    ?(ACT1 (ISA-INSTANCE-OF POSSESS-INANIMATE-OBJECT $ACT1))
    ?PLAYER1 ?LOCATION1 ?OBJECT1)

  SECONDARY-ENABLING-CONDITION

   (?(TIMEØ (JUST-BEFORE $TIMEØ $TIME2))
    ?(ACTØ  (OR (ISA-INSTANCE-OF LOCATE-ANIMATE-OBJECT $ACTØ)
                (COULD-COUNT-AS-A-INSTANCE-OF LOCATE-ANIMATE-OBJECT $ACTØ))
    $PLAYER2 $LOCATION2 $OBJECT2)

  PRIMARY-CONSEQUENCES

    (AND
      (?(TIME3 (JUST-AFTER $TIME3 $TIME2))
       ?(ACT3 (ISA-INSTANCE-OF MOVING-INANIMATE-OBJECT $ACT3))
       $OBJECT2 ?LOCATION3 ?MODIFIERS)
      (?(TIME4 (AFTER $TIME4 $TIME3))
       ?(ACT4 (ISA-INSTANCE-OF LOCATE-INANIMATE-OBJECT $ACT4))
       ?PLAYER4 $OBJECT2 ?LOCATION4))

  SECONDARY-CONSEQUENCES

   (?(TIME5 (JUST-AFTER $TIME5 $TIME2))
    ?(ACT5 (OR (ISA-INSTANCE-OF LOCATE-ANIMATE-OBJECT $ACT5)
               (COULD-COUNT-AS-A-INSTANCE-OF LOCATE-ANIMATE-OBJECT $ACT5))
    $PLAYER2 ?LOCATION5)

  SKILL-ENERGY-REQUIREMENTS

    (FAST MOVING-INANIMATE-OBJECT (HIGH ENERGY) (HIGH SKILL))
    (SLOW MOVING-INANIMATE-OBJECT (LOW ENERGY) (MEDIUM SKILL))

  INCREASED-SKILL-ENERGY-CAN-PRODUCE

    (FASTER MOVING-INANIMATE-OBJECT (HIGH ENERGY) (HIGH SKILL))
    (FARTHER MOVING-INANIMATE-OBJECT (HIGH ENERGY) (HIGH SKILL))
```

Figure III.3 Act Schema for Action THROW

The basic aspects of an action are described in an act schema: preconditions and outcomes. Effectively, such a description defines the relationships between actions. Note also in the above description of a THROW, that the ENABLING-CONDITIONS and CONSEQUENCES are stated in terms of the primitive action types, and not in terms of specific, observable actions. In addition, an act-schema contains information about the performance of an action: its skill and energy requirements and the variability in possible outcomes. Recall the interpretation for the notation: names prefixed by a ? indicate an open variable, while names prefixed by a $ indicated a closed variable (one that must have been assigned a value previously); names not prefixed are constants.

rather than in terms of the observable actions. The PRIMARY-CONSEQUENCE for THROW states that the object THROWn, first must be MOVING and, sometime later that the same object must LOCATE itself at some location. This specification does not state how the ball should be moving -- flying in the air, rolling on the ground, etc. Nor does this specification state how the object will come to be at a location -- whether by being caught, or simply by rolling to a stop, etc.

We define a causal enablement chain to be a sequence of actions in which the consequences of one action (or possibly several actions) satisfy the preconditions of a subsequent action. Let us now see how the specification of the enabling conditions and the consequences permits the system to tie together observed actions in this way. At the bottom of Figure III.4 we see the oft repeated snapshot sequence of [pitcher THROWs BALL -- batter HITs BALL]. The objective is to discover that the pitcher's THROW (snapshot 102) satisfied the physical enabling conditions for the SWINGHIT (snapshot 104). At the top of Figure III.4 we see abbreviated versions of the act-schemas for THROW and SWINGHIT. Assume that the system has just observed the action SWINGHIT and it is now trying to reconstruct what actions enabled that act. First, the system would access the PRIMARY-ENABLING-CONDITIONS for the observed action SWINGHIT. Then it would begin to match those conditions against the actions which just occurred.

In Figure III.4 we see that the primitive action MOVING-INANIMATE-OBJECT in the precondition clause of SWINGHIT is successfully matched

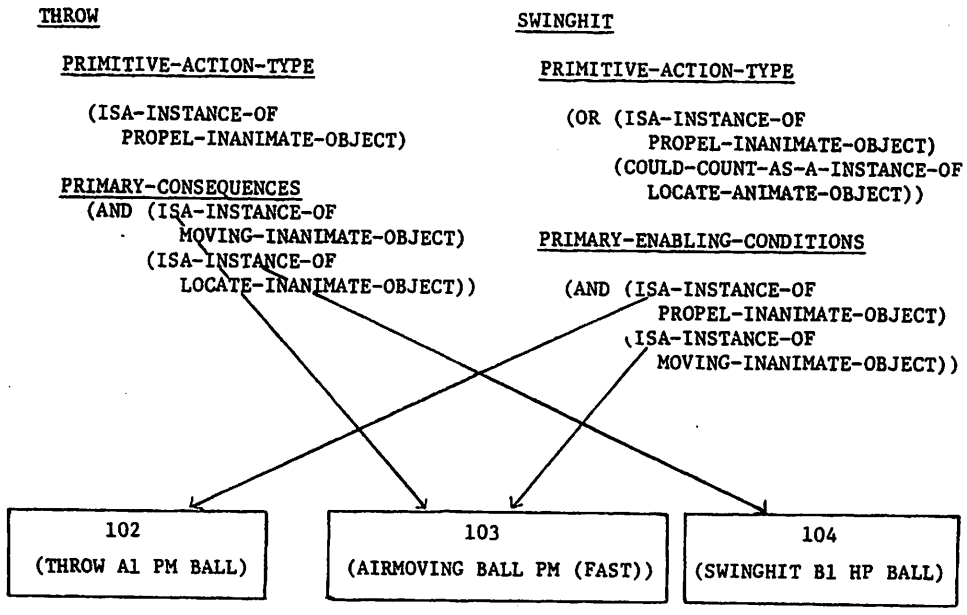


Figure III.4 Using Act-Schemas to Infer Causal Enablement Relationships

In this figure we see how act-schemas can be used to establish causal links between actions; the PRIMARY-CONSEQUENCES of A1's THROW satisfy the PRIMARY-ENABLING-CONDITIONS for B1's SWINGHIT.

against the action AIR-MOVING BALL in the observations (snapshot 103), since AIR-MOVING BALL ISA-INSTANCE-OF MOVING-INANIMATE-OBJECT. Next, the PROPEL-INANIMATE-OBJECT is successfully matched against the THROW by player A1 (snapshot 102), since THROW ISA-INSTANCE-OF PROPEL-INANIMATE-OBJECT. Similarly, the PRIMARY CONSEQUENCES from the THROW act-schema are successfully matched against the observations in snapshots 103 and 104. The results of this process establish a causal enablement chain between the action THROW in snapshot 102 and the action SWINGHIT in snapshot 104.

All act-schemas are implemented as demons [SEL59]; when triggered, they attempt to establish causal enablement chains. In so doing, they have the capability of augmenting the pattern descriptions of the participating actions. Thus in the above example, the act-schema for THROW adds the feature (ENABLED(104 SWINGHIT)) to the THROW pattern description, while the act-schema for SWINGHIT adds the complimentary feature (ENABLED-BY(102 THROW)) to the SWINGHIT pattern description (Figure III.5).

A causal enablement chain can also be established using the secondary consequences of an action. Consider the snapshots at the bottom of Figure III.6; in snapshot 104 the player B1 HITS the BALL and in snapshot 105 the same player begins to RUN. At the top of Figure III.6 we include a partial description of the act-schemas for SWINGHIT and RUN. A causal enablement chain can be established between these two actions in the following way. First, the PRIMARY-ENABLING-CONDITIONS for the action RUN

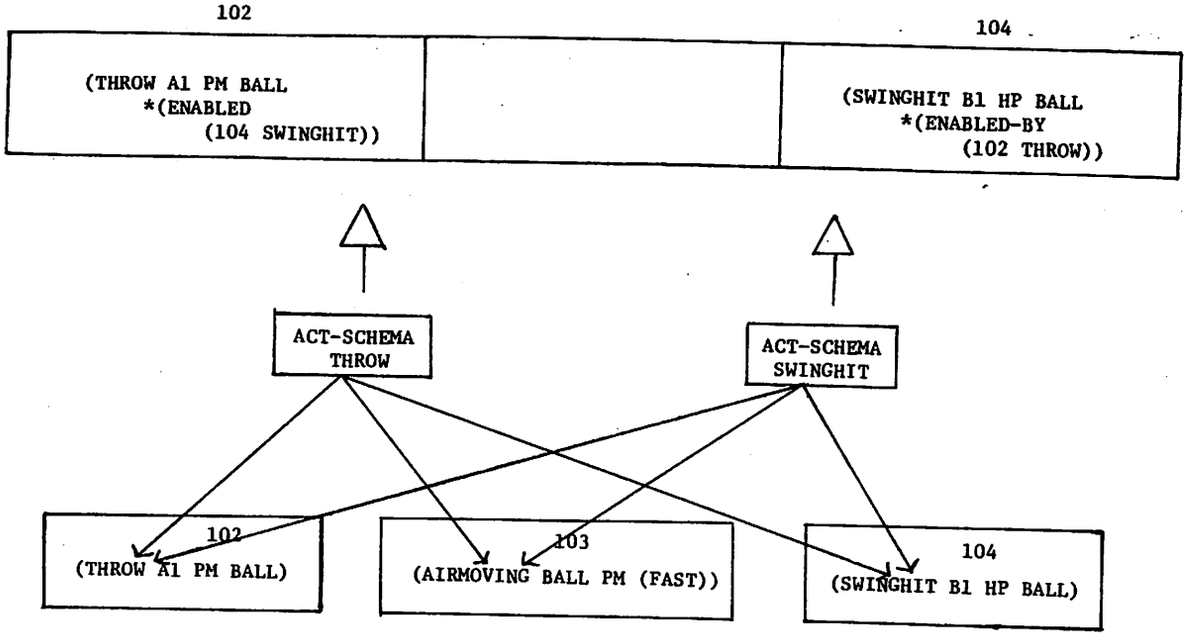


Figure III.5 Adding New Features to Pattern Descriptions

The effect of triggering an act-schema and establishing a relationship is the augmentation of the appropriate pattern descriptions with new features to reflect this inference. The features asterisked in this figure are added to the pattern descriptions of THROW and SWINGHIT to reflect the inference of A1's enablement of B1's action.

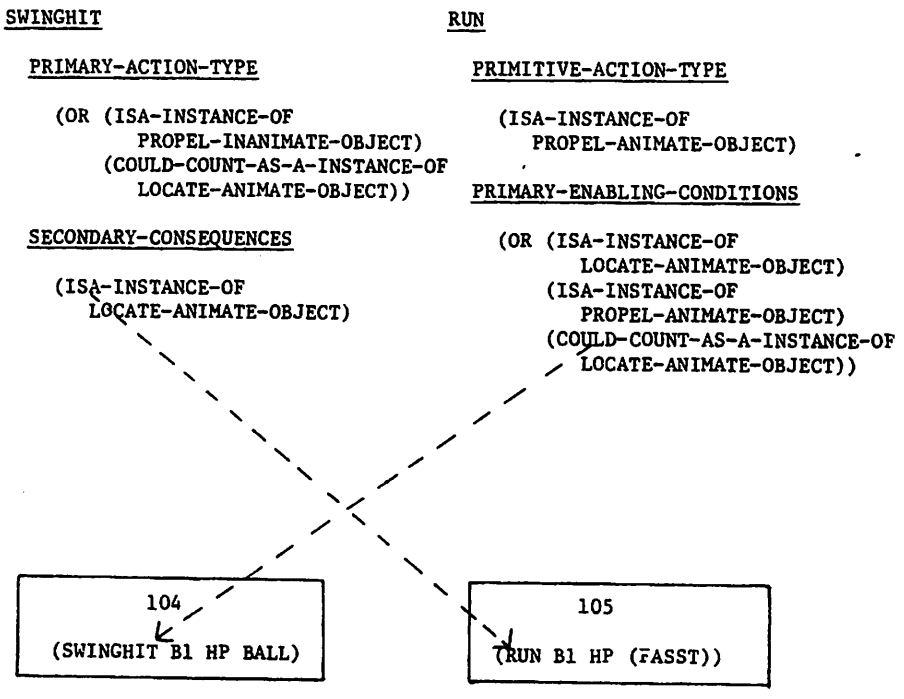


Figure III.6 Establishing a Causal Enablement Chain Using the SECONDARY-CONSEQUENCES of an Action

In trying to fix the location of B1 before he started RUNning, it seems reasonable to view B1's SWINGHIT as a type of LOCATE-ANIMATE-OBJECT action. This fact is reflected in the second clause under the indicator PRIMARY-ACTION-TYPE in the act-schema for SWINGHIT, and in the SECONDARY-CONSEQUENCES listed for that action. This fact also explains the existence of the third clause under the indicator PRIMARY-ENABLING-CONDITIONS in the act-schema for RUN.

are fetched up. They state that the enabling condition for RUN must be a preceding action which either ISA-INSTANCE-OF LOCATE-ANIMATE-OBJECTs, (e.g., AT, ON), or an action which ISA-INSTANCE-OF PROPEL-ANIMATE-OBJECTs, or an action which COULD-BE-AN-INSTANCE-OF LOCATE-ANIMATE-OBJECTs. The first two clauses of RUN's enabling condition cannot be matched against the preceding observed SWINGHIT, since its primary consequences do not propel or locate animate objects. However, if we look at secondary consequences of SWINGHIT, we see that the effect of propelling the BALL does not constrain what could happen to the player. The hitter might either continue to stand at the current location (LOCATE-ANIMATE-OBJECT), or he might not (PROPEL-ANIMATE-OBJECT). Thus, it seems fair to say that a SWINGHIT COULD-BE-AN-INSTANCE-OF LOCATE-ANIMATE-OBJECTs, if we are only concerned with the status of the player's location and not the ball. The third clause of RUN's enabling condition can now be successfully matched against the preceeding SWINGHIT.

Intuitively, however, the strength of connection between the SWINGHIT and the RUN of Figure III.6 is not equal to the strength of connection between the THROW and the SWINGHIT of Figure III.4. In the latter case the two actions are linked tightly together by physical causality, whereas in the former case there seems to be an incidental flavor to the relationship. We indicate this difference in connection by using dotted lines to indicate the links between the SWINGHIT and the RUN (Figure III.6). In Figure III.7 the features added to the pattern descriptions of these actions also reflect this difference; we say that the

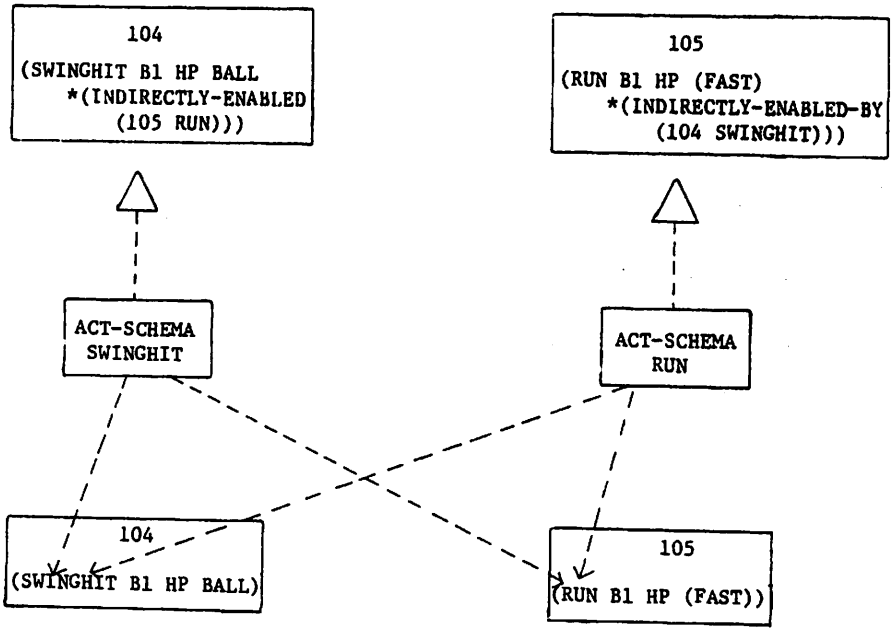


Figure III.7 Direct Enablement vs Indirect Enablement

The SECONDARY-CONSEQUENCES of an action capture the more non-necessary (incidental) aspects of an action. A causal relationship based on such consequences reflects a weaker relationship than one between the PRIMARY-CONSEQUENCES of an action. This difference in connection is indicated by dotted lines between the act-schemas and in the feature INDIRECTLY-ENABLED and INDIRECTLY-ENABLED-BY which is added to the pattern descriptions.

SWINGHIT INDIRECTLY-ENABLED the RUN and correspondingly, the RUN was INDIRECTLY-ENABLED-BY the SWINGHIT. In the discussion of cooperation among players of the same team, we will make use of this distinction (section IV.3.3).

The above example illustrates how the primitive actions are used to define the enabling conditions and the consequences of an observable action. Rather than encoding in each observable action all the alternative observable actions that could take place, we encode this variability information once and for all in the description of the primitive actions. If new observable actions were added to the system -- those outside the game of baseball -- the same act-schemas currently employed would not need to be updated, since they do not refer directly to the observable actions but rather indirectly to them through the primitive actions.

III.2.3 The Frame Problem Attempts to Rear Its Ugly Head

The careful reader will have noticed that the primary enabling conditions and primary consequences for the actions THROW and SWINGHIT only dealt with a single object, the BALL. What became of the person who THREW the BALL or the person who HIT the BALL after the execution of those actions? Consider the following analogous problem (taken from [SRI76]): at time t_0 a man is standing in the kitchen holding a full cup of coffee, at time t_1 he walks into the living room, and at time t_2 he is standing in the living room. Question: where is the cup of coffee? We know, that barring some unforeseen happening, the coffee cup is still

with the man at time t_3. This is so because the action "walking-person" implies that whatever a person possesses (e.g., his clothes) at the initiation of the action continues to be in his possession unless explicitly advised to the contrary. In general, a system must have a description of the important side-effects of actions. The problem of specifying such effects and updating a model of the world to reflect the changes caused by the side-effects has been called the Frame-Problem [McC69, SRI76].

Our approach to coping with the Frame Problem is based on the assumption that, taken together, the primary and secondary consequences of an action specify all the effects that action has on the world. Thus, in updating its model of the physical interactions of the world, our system simply checks that all the consequences are satisfied. Implicit in this procedure is the assumption that those events not explicitly referenced by an act-schema do not change. This has been called the STRIPS assumption [WAL77].

This technique is sufficient for our purposes since the amount of activity is low in the games of baseball which our system observes. Moreover, we can specify all the important changes which will occur in that domain. Thus, there is no question as to the correctness of our system's interpretation at this level; what the system thinks has happened, did, in fact, occur. However, in a more complex domain, it may not be possible to specify all the possible interactions nor may it be computationally efficient to trace down on all the dependencies. A system might

have to adopt a strategy that its model of the interactions is only partially correct; only when a discrepancy between the world and the model arises are more sophisticated analysis procedures brought to bear [SRI76]. The acceptability of either approach seems to depend on the domain and on the requirements of the system.

III.2.4 Declarative Knowledge in Act-Schemas

As we said earlier, those aspects of the act-schemas which make inferences about the relations between actions are implemented as procedures. This information was used for its effect -- the augmentation of the pattern descriptions during the first phase of hypothesis Generation. However, additional knowledge about other aspects of actions will be needed during the domain-specific interpretation phase of Hypothesis Generation. Since this additional knowledge will be "read" by other procedures and not executed for effect, it can be called declarative knowledge.

The first type of descriptive information included in the act-schema of an action deals with the degree of skill and energy required to perform the particular action. Consider the abbreviated act-schema for THROW in Figure III.8. Under the indicator SKILL-ENERGY-REQUIREMENTS, we see that THROWing a BALL to a location at a high velocity requires a goodly amount of skill and energy, while THROWing a BALL at a low velocity requires only modest amounts of skill and energy. While these measures are coarse, they are sufficient for our current purposes. For example, consider again the 3 snapshot sequence depicted in Figure III.1. In hypothesizing that player A1 is involved in a competitive relationship with B1, it will be

```
THROW

  SKILL-ENERGY-REQUIREMENTS

    (FAST MOVING-INANIMATE-OBJECT
       (HIGH ENERGY)
       (HIGH SKILL))
    (SLOW MOVING-INANIMATE-OBJECT
       (LOW ENERGY)
       (MEDIUM SKILL))
```

Figure III.8 Skill and Energy Requirements

Declarative information is information which is read by other procedures, and not executed for effect. The knowledge that "to THROW a BALL at a high rate of speed requires a high degree of skill and energy" is expressed as declarative information in the act-schema for THROW; procedures will access this information in attempting to hypothesize goals for players engaged in acts of THROWing.

```
RUN

  INCREASED-SKILL-ENERGY-CAN-PRODUCE

    (FASTER MOVING-ANIMATE-OBJECT)
    (FARTHER MOVING-ANIMATE-OBJECT)
```

Figure III.9 Variability in Outcomes

Also expressed as declarative information is the knowledge about the changes which can be brought about by an actor in performing an action with a greater amount of skill and energy. We see here that with an increase in skill and energy a player can RUN FASTER and/or FARTHER.

important to know if the THROW and the SWINGHIT are 'difficult actions'; i.e., one which requires a medium-to-high degree of skill and/or energy. The system determines that the THROW by player A1 in Figure III.1 is a 'difficult-act' by matching the observed activity (the BALL MOVING FAST) with information under the SKILL-ENERGY-REQUIREMENTS indicator (THROWing a FAST MOVING BALL requires a HIGH degree of SKILL and ENERGY) of the act-schema for THROW. The system uses a similar analysis to decide that the SWINGHIT by player B1 in Figure III.1 is also a 'difficult-act'.

The second type of descriptive knowledge in each act-schema deals with possible variations in the primary consequences of an action and the corresponding increases in skill and energy needed to bring about those variations. Figure III.9 illustrates an abbreviated act-schema for RUN. The clauses after the indicator INCREASE-SKILL-ENERGY-CAN PRODUCE state that an increase in the amount of skill and energy used in the execution of a RUN can permit an individual to RUN FASTER or FARTHER (during any given interval of time). There is a procedure which accesses this information in order to reason hypothetically about possible alternative outcomes. For example, it will be important to know if an individual RUNning can cover the distance to a location in a shorter period of time. Using the information that an individual could RUN FASTER in conjunction with an understanding of general space-time relationships, this procedure would decide that such a situation was possible.

III.3 Summary

We conclude this chapter with the following observations: the system's common-sense knowledge of the physics of actions is used during the first phase of Hypothesis Generation. It provides an interpretation of the observed activity in terms of spatio-temporal relationships; that interpretation results in the augmentation (addition of features) of the pattern descriptions of the participating actions. We can call this the non-intentional level of interpretation, since no reference to goals has been made.

The representations employed in this chapter illustrated an interesting mix of procedural and declarative knowledge. It seems that these terms can best be understood with reference to the use to which the encoded knowledge is put rather than to its implementation; if the encoding is executed for its effect, then it is procedural; if it is "read" by other procedures, then it is declarative.

We have had to employ a great deal of machinery in order to produce the desired analysis in the relatively simple spatio-temporal domain of baseball. Clearly, we have only tapped the surface -- a child surely has a richer and more robust common-sense understanding of the physics of actions.

C H A P T E R I V

CONSTRUCTING AN INTERPRETATION: UNDERSTANDING THE NEW IN TERMS OF THE OLD

What is the structure of a competitive action game? What general rules must a system have in order to interpret the observed actions as a game? How is this knowledge used to infer the goals and relationships of players? In this chapter, we shall attempt to answer these questions by first providing a detailed account of the domain knowledge provided to the system. Then we shall explore the contribution of that knowledge on system performance. Finally, we shall survey some of the work by other researchers in the area of interpretive system.

IV.1 Content of Old Knowledge: A General Description

People seem to paraphrase, summarize, or in general describe, the activity of other people[10] using terms such as purposes, goals, needs, desires, wants, and intentions. Consider the story taken from [SCHM77a] in Table IV.1 and some summaries collected from subjects who were asked to describe what happened in the story (Table IV.2, also taken from [SCHM77a]). Notice first that some subjects (summaries a, b, c) in their descriptions used terms such as "wants" which were not in the story. Moreover, subjects recognized that "Tom" wanted something to eat and thus was trying to make an ice cream cone, even though this fact was not mentioned in the story -- and even though the event never happened!

Tom walks into the kitchen.
Tom opens the cabinet.
Tom takes a box of cones out of the cabinet.
Tom takes a cone out of the box.
Tom places the cone on the cabinet.
Tom closes the box of cones.
Tom places the box of cones in the cabinet.
Tom closes the cabinet door.
Tom walks to the refrigerator.
Tom opens the door of the freezer.
Tom closes the freezer door.
Tom walks to the cabinet.
Tom opens the cabinet door.
Tom puts the cone in the box.
Tom closes the cabinet door.

Table IV.1 A Simple Story

In this story, Tom attempts to make himself an ice cream cone. This story is taken from [SCHM77a].

(a)

Tom thought
he wanted an ice cream cone
so he gets his cone.
Then when he goes
to get the ice cream
he decides
he doesn't want it
or
that there's no ice cream
so he puts the cone back into the box.

(b)

Tom wanted some ice cream
but he found
that there was none.
Tom is very neat.

(c)

Tom fails
in getting the ice cream cone
he wants.

(d)

Tom walked into the kitchen
and took out a box of cones out of the cabinet.
He closed the box of cones
put it on the cabinet
and then put it into the cabinet
and closed the cabinet.
He then walked to the refrigerator
opened the freezer,
closed the freezer,
opened the cabinet,
took out the box of cones,
opened the box,
put the cone back in the box,
put the box in the cabinet
and closed the cabinet.

Table IV.2 Story Summaries

Subjects were asked to summarize the story depicted in Table V.1. Above, we list four such summaries. Notice that the first three (a, b, c) introduce the purposive term 'want' and identify Tom's goal -- neither of which were originally in the story. The fourth summary -- if one would even call it a summary -- simply describes the events in the story (also taken from [SCHM77a]).

We call a description which uses such purposive terms as goal, intention, or want, a "common-sense description". Another type of description, which could be called non-intentional, would use terms such as those that were in the story itself (e.g., summary d).

These observations have prompted many researchers (e.g., [ARB72, MIN75,SCHA77, SCHM76a] to posit the existence of knowledge structures and processes in the minds of the observers which serve to add the purposive terms to non-intentional descriptions. People must have a stored model (or models) of human behavior which is expressed both in terms of the observable actions, and the goals, plans, etc. which are facilitated by those actions. Using such a model an observer can infer possible goals for an individual and his actions, based on the type and outcome of the interactions with the environment. For example, from the type of Tom's interactions with the objects in the kitchen and the outcome of his walking empty-handed from the kitchen, we were able to infer that Tom wanted something to eat and that Tom was not able to make himself an ice cream cone. Similarly, we have initially provided our system with a model of competitive action games consisting of a set of rules which characterize the kinds of interactions often found in games, and the goals often implied by those interactions. We call such interactions "competitive or cooperative causal relationships", and the rules which capture such knowledge "Causal-Link Schemas" (CLSs).

There is another important observation which can be drawn from the summaries of the story presented earlier. Namely, the subjects summarized

a whole set of actions and subgoals by supplying the major, or final, intended goal: Tom wanted to eat an ice cream cone. The implication is that purposive behavior is structured; it seems that local subgoals hierarchically compose plans [SCHM76b, NEW59].

Baseball is an example of goal-directed behavior, and as such, it too exhibits the above goal structuring property. Opposing teams compete with each other to win the game -- the final goal, while opposing players compete with each other down at the level of individual interactions -- the pitcher versus the batter. Thus, not only is our learning system equipped with knowledge about local competitive interactions and goals (the CLSs), but it also can compose local goals into hierarchically structured plans.

IV.2 Annotated Example of Causal-Link Schema Application

The system is in the position of an observer who has just seen some action by a player. It seeks to explain why that action took place by identifying: (1) a goal (or goals) which the player was attempting to achieve; (2) a competitive and/or cooperative relationship (or relationships) with an action (or actions) performed by a teammate or opponent, which occurred in close temporal proximity to the action under investigation. If found, we say that the latter action "enabled" the former action. Words fail us at this point. By "enabled" we do not mean physically-enabled -- those inferences were already made at the non-intentional level of the act-schemas. Here, "enabled" means "competitive-enable" or "cooperative-enable" at the logical level of game relationships.

The specific kind of competitive relationship we wish to examine in this example could be termed PHYSICAL-COMPETITION; a player on one team physically enables the action of an opponent. The interaction between the pitcher A1 and the batter B1 in Figure IV.1 is an instance of such a competitive interaction. Let's see how our system discovers this relationship in the above example.

Consider the Causal-Link Schema (CLS) in the top-half of Figure IV.1. In the form of production rules [NEW73], CLSs ask a series of questions about properties of the observed actions. If all the questions turn out to be true, then the clauses on the right-hand side of the IF statement can be instantiated, i.e., hypotheses asserted. As we shall see, the particular CLS depicted in Figure IV.1 recognizes what might be termed PHYSICAL-COMPETITION.

Now, assume that the system just observed player B1's SWINGHIT in snapshot 103 of Figure IV.1 and is in the process of constructing an explanation for that action in terms of some competitive relationship. Since A1's THROW (snapshot 101) occurred shortly before B1's SWINGHIT (snapshot 103), the former is taken as a candidate to be examined further. Thus, the variables in the predicates of the CLS in Figure IV.1 can be bound to the following values: the pattern description for A1's THROW can be bound to the variable ACT1 and player A1 can be bound to PLAYER1; similarly, the pattern description for B1's SWINGHIT can be bound to ACT2 and player B1 bound to PLAYER2. Finally, assume that the act-schemas for THROW and SWINGHIT have made an inference about the

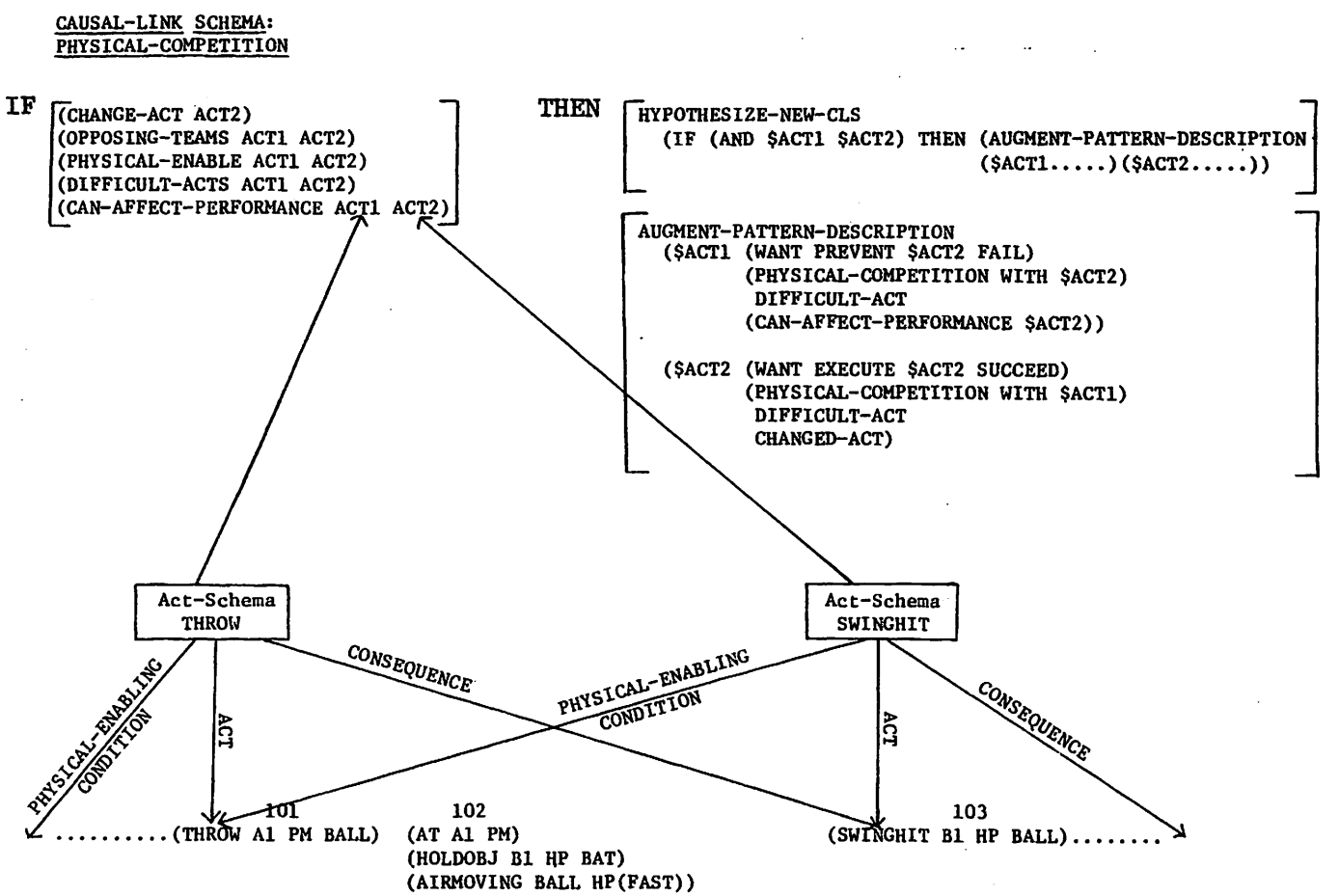


Figure IV.1 Application of a CLS to Observed Actions

The Causal-Link Schema PHYSICAL-COMPETITION is used here to hypothesize a competitive interaction between the pitcher A1 and the batter B1. This CLS, like the nine others, are represented as production rules. The left-hand side, the condition part, tests for various properties of the actions. If all the tests return true, which they do in this case, then the right-hand side can be triggered. Notice that two actions result: a production rule is hypothesized, which is particularized for the interaction at hand and the pattern descriptions for the appropriate actions are augmented with new features such as goal and competitive relationship.

physical relationship between these two actions; the THROW by A1 ENABLED the SWINGHIT by B1.

The first predicate on the left-hand side of the IF statement is (CHANGE-ACT ACT2). In goal-directed behavior, a change in a participant's actions often indicates a change in goals. Moreover, such changes might be due to some relationship with an opponent or with a teammate. At a minimum then, explanations for changes in a player's actions must be sought. Currently, the system does not attempt to explain why a player continued to perform some action. Returning to our example, since B1's previous action was simply to stand and HOLD a BAT at HOMEPLATE (snapshot 102), this predicate returns TRUE.

The system can now proceed to the second test, (OPPOSING-TEAMS ACT1 ACT2). If B1's change in actions is to be explained in terms of a competitive relationship, the candidate player must be on the opposing team. The second question determines the membership status of the players in the candidate interaction; A1 is on the A-team while B1 is on the B-team, thus this predicate is also satisfied.

The third predicate, (PHYSICAL-ENABLE ACT1 ACT2), seeks to establish the direct physical connection between the two actions in question. By accessing information in the pattern descriptions for these actions, this predicate discovers that the act-schemas have inferred that a physical enabling relationship does exist between the THROW and the SWINGHIT, and thus can also return true.

Just because B1's action was enabled by A1's action does not mean that B1 intended or wanted to execute that action. Similarly, we still do not know if A1 did or did not want to enable B1's action. Judgement about these issues requires that a natural assumption about games be made: a player does not unintentionally perform an action which requires a high degree of skill and energy. HITting a BALL, which is moving FAST with a stick-like object is an action which does require a significant degree of skill. It is unlikely that a player would execute such an action if he had not intended to do so. This observation stems from the very nature of competitive action games; highly skilled players perform physical actions which test the limits of their physical abilities. Moreover, actions which require a significant degree of skill and/or energy are often the important ones in a game. Moreover, we assume that a player would not usually want to enable the execution of such an action by an opponent. For lack of a better term, we define "difficult-act" to be an act which requires a high degree of skill and/or energy for performance. By accessing information in the appropriate act-schemas as to the amount of skill and energy required to perform a THROW and a SWINGHIT, the fourth predicate (DIFFICULT-ACTS ACT1 ACT2) determines that both of the actions can be considered to be "difficult-acts".

Implicit in the nature of a competitive game is the notion that both participants (players or teams of players) in a competitive event have a chance at winning, i.e., succeeding at achieving some distinguished goal. This property is also true of the interaction between two opposing players.

The fifth predicate (CAN-AFFECT-PERFORMANCE ACT1 ACT2) attempts to establish that this property is true in the candidate competitive interaction. It attempts to determine if the complimentary alternative to the observed outcome of the interaction could have taken place, i.e., the above predicate reasons hypothetically to determine if the pitcher could have THROWn the BALL towards the batter in such a way as to decrease the likelihood of player B1 being able to execute his SWINGHIT. In our example, this amounts to examining the relationship of THROWing the BALL and subsequently HITting the BALL.

Question: What effects could occur if the pitcher A1 had applied an increased amount of skill and energy to the performance of his THROW?

Answer (from information contained in the act-schema for THROW): The BALL could travel FARTHER and/or the BALL could travel FASTER.

Question: What effects could the above conditions have on the performance of SWINGHIT?

Answer (from information contained in the act-schema for SWINGHIT): It would have been more dificult, hence less likely, to HIT a BALL which was moving FASTER.

Thus it is possible that the pitcher could have THROWn the BALL, and the batter might not have HIT it. If there were nothing the pitcher could have done -- short of <u>not</u> THROWing the BALL at all or THROWing it completely out of B1's range -- which would have had some negative effect on B1's SWINGHIT, one would probably not call such an interaction "competitive". As we have seen then, this predicate can also return true.

Since all the questions in the left-hand side of the CLS of Figure IV.1 are true, hypotheses on the right-hand side of the CLS can be triggered. The inference made in this case is that a competitive relationship, PHYSICAL-COMPETITION, seems to exist between A1 and B1. Moreover, based on the preceding argument that players in a competitive game do not usually intend to enable a "difficult-action" of an opponent, A1's goal was hypothesized to be that of preventing his opponent's action of SWINGHIT, and thus he FAILed with his goal in this instance. Similarly, based on the assumption that players usually intend to execute difficult-acts, B1's goal was hypothesized to be that of wanting to execute the action SWINGHIT, and thus he SUCCEEDed with his goal in the observed instance.

In Figure IV.1 we see on the right-hand side of the IF statement the function AUGMENT-PATTERN-DESCRIPTIONS which takes as arguments the actions and the specific goal and competitive relationship hypotheses, and the features computed in the CLS. Figure IV.2(a) depicts the effects of executing that procedure for our example; the above values for the features "goal" and "competitive causal relationship" have been added to the pattern descriptions of A1's and B1's actions. Values for the features "change-act", "difficult-act" and "can-affect-performance" are also added. (Recall that the physical-enable relationship has already been added by the appropriate act-schemas.)

One of the important tests of a learning system is its ability to <u>use</u> the knowledge that it acquires. This objective requires that the

AUGMENTED PATTERN DESCRIPTIONS

```
(101 THROW A1 PM BALL                                  (103 SWINGHIT B1 HP BALL
     (ENABLED (103 SWINGHIT))                               (ENABLED-BY (101 THROW))
     (WANT PREVENT (103 SWINGHIT) FAIL)                     (WANT EXECUTE (103 SWINGHIT) SUCCEED))
     (PHYSICAL-COMPETITION WITH (103 SWINGHIT))             (PHYSICAL-COMPETITION WITH (101 THROW))
      DIFFICULT-ACT                                          DIFFICULT-ACT
      (CAN-AFFECT-PERFORMANCE (103 SWINGHIT)))               CHANGED-ACT)
```

(a)

NEW PRODUCTION RULE/CAUSAL-LINK SCHEMA

```
IF (AND                                          THEN (AUGMENT-PATTERN DESCRIPTION
     (101 THROW A1 PM BALL                              (101 THROW A1 PM BALL
          (ENABLED (103 SWINGHIT)))                          (ENABLED (103 SWINGHIT))
     (103 SWINGHIT B1 HP BALL                                (WANT PREVENT (103 SWINGHIT) FAIL)
          (ENABLED-BY (101 THROW))))                         (PHYSICAL-COMPETITION WITH (103 SWINGHIT))
                                                              DIFFICULT-ACT
                                                             (CAN-AFFECT-PERFORMANCE (103 SWINGHIT)))

                                                        (103 SWINGHIT B1 HP BALL
                                                             (ENABLED-BY (101 THROW))
                                                             (WANT EXECUTE (103 SWINGHIT) SUCCEED))
                                                             (PHYSICAL-COMPETITION WITH (101 THROW))
                                                              DIFFICULT-ACT
                                                              CHANGED-ACT))
```

(b)

Figure IV.2 The Result of Triggering a CLS: Augmenting Pattern Descriptions and Production of a New CLS.

In (a) we see that the CLS PHYSICAL-COMPETITION augmented the pattern descriptions for THROW and SWINGHIT to reflect their hypothesized competitive relationship. In (b) we see that a new CLS was produced which captured the same information that was in (a). The active production rule form (when suitably generalized) will be used to recognize subsequent examples of this interaction.

acquired knowledge be in a representation which is executable by the system. However, the pattern description representation is a "passive one" -- it is simply a set of features. Therefore, in addition to augmenting pattern descriptions, CLSs generate production rules, which capture in a procedural way, the same information encoded in the pattern descriptions. On the right-hand side of the CLS depicted in Figure IV.1, the function HYPOTHESIZE-NEW-CLS serves to generate the new production rule when the CLS is triggered. Effectively, general CLSs produce specific CLSs; once verified (see Chapter VII) the acquired CLSs can be used in the same way as the general CLSs which were initially provided. Figure IV.2b depicts the specific CLS with its variable arguments instantiated which was hypothesized by the general CLS PHYSICAL-COMPETITION in our example.

CLSs hypothesize "local" goals. In the above example, it was hypothesized that B1's goal was to execute the SWINGHIT. However, the question still remains: why did B1 want to execute that action? Either executing that action is an end in itself and is thus desirable, or executing that action is desirable because it facilitates a subsequent action-goal. In the introduction to this chapter we pointed out that goals were structured hierarchically into plans. Therefore, in order to answer the above question, the "local goal hypothesis" must be integrated into a global plan. Section IV.3.5 discusses how this latter process is performed.

Let us close this section by summarizing some of the issues raised in the example. We have seen how general stored knowledge can be used to recognize, and thus assimilate new information into the system. As it stands, the acquired knowledge is particularized general knowledge. We shall see later how this new knowledge is generalized so as to be applicable in a wider range of contexts. Also, we shall see that multiple hypotheses suggesting conflicting interpretations are often made. Chapter VII will describe how the system copes with this problem. Finally, we have discussed how acquired knowledge can subsequently be used in the system, which is afterall the ultimate test of a learning system. With that, let us now plunge into a detailed discussion of the above issues.

IV.3 A First-Order Characterization of Competitive Action Games

In order to interpret the observations, the system needs some technical knowledge about games. Where is this knowledge to come from? Following the "knowledge-base" paradigm outlined in Chapter I, we should go out and find experts in the field of games. Unfortunately, it is not as simple as that.

There <u>is</u> a diverse literature on games (see [AVE71a] for an extensive bibliography). Anthropologists, sociologists, psychologists, and mathematicians have studied games from a wide range of perspectives and purposes, e.g., to isolate the invariant structures in all games, to understand the implications for human development of this "universal

grammar" of games, to understand the impact of game playing on personality, to develop good techniques for teaching games, to develop a mathematical characterization of games. Our goal, however, is somewhat different; namely, describe in general terms, the common-sense knowledge about games which would be sufficient to enable a system to learn about a particular game. While some of the above work is relevant to our objective (e.g., [REI71, AVE71b]), the discussions in the game literature do not address the issues we are interested in at a level of specificity which a literal, unforgiving computer requires.[11] Thus, we have had to become our own domain experts in order to define concepts about games in rule form which are precise enough for a computer.

IV.3.1 Competition and Cooperation

The rules which we have developed are based on a characterization of action games in terms of two basic concepts: competition and cooperation. Generally speaking, competition can be defined as interactions between players on two teams,[12] where players on team-A try to prevent players on team-B from achieving their goals, and vice versa. Cooperation can be defined as interactions between players on the same team, where those players try to help each other achieve some common goal (or goals). "Winning a game" occurs when team-A achieves some distinguished goal(s), while preventing team-B from also achieving the distinguished goal(s), or vice versa.

In the following sections, we will first show how the general definitions of competition and cooperation are applicable down at the level

of competing and cooperating individuals. Next, we will discuss how this characterization of competition and cooperation is used to move up the goal hierarchy by integrating local goals into more global episode-plans. Let the reader be under no false illusions; we have not been able to characterize all the levels of goals and plans in a game -- from individual interactions to winning and losing. The stage, however, has been set. Finally, while the wealth of details may seem "unimplementable", we would like to assure the reader that all that follows (unless explicitly noted) has been implemented and is running. In Chapter VIII we document some experiments performed with the system.

IV.3.2 Local Competitive Interactions

In a competitive game typically one team wins and the other team loses. We might formalize that fact as follows:

(1) GAME-COMPETITIVE-INTERACTION(ACTS-OF(TEAM1),ACTS-OF(TEAM2))→
(FAIL(GOAL-OF(TEAM1)) ∧ SUCCEED(GOAL-OF(TEAM2))) ∨
(SUCCEED(GOAL-OF(TEAM1)) ∧ FAIL(GOAL-OF(TEAM2))) ∨

where

GOAL-OF(TEAM1) = ACHIEVE-DISTINGUISHED-EVENT ∧
PREVENT(GOAL-OF(TEAM2)),

and vice versa; "∨" is an exclusive or.

This property is crucial in defining the concept of a competitive game, but it is not necessarily true of other types of social interactions. In fact, Piaget [PIA65] has observed that young children do not have the above rule, i.e., they think that both teams or players can win! Two simplifying assumptions have been made in this characterization:

(1) games in which "ties" (neither team wins) are excluded; (2) games in which the distinguished goal is not symmetric have also been excluded.

A goal structure similar to the one for games also seems to hold for local competitive interactions; one player succeeds or wins while an opposing player fails or loses. We might say that

(2) LOCAL-COMPETITIVE-INTERACTION(ACT-OF(PLAYER1),ACT-OF(PLAYER2))→
(FAIL(GOAL-OF(PLAYER1)) ∧ SUCCEED(GOAL-OF(PLAYER2))) ∨
(SUCCEED(GOAL-OF(PLAYER1)) ∧ FAIL(GOAL-OF(PLAYER2)))

where PLAYER1 and PLAYER2 are on OPPOSING TEAMS, and
where the "∨ " is an exclusive or.

For example, in Section IV.2, it was hypothesized that the pitcher and batter (Figure IV.1) were in the competitive relation PHYSICAL-COMPETITION, with the pitcher FAILing and the batter SUCCEEDing with their respective goals.

Immediately, however, apparently anomalous situations with respect to (2) arise. In football, for example, it would seem that both teams FAILed in the situation where a "pass is not completed" (a teammate of the quarterback fails to catch the football and the opposing team did not intercept the pass). However, a more careful analysis of this situation requires that several <u>levels</u> of goals be distinguished. The passing team FAILed by not completing the pass, but SUCCEEDed by not losing possession of the ball. The opposing team FAILed by not intercepting the pass, but SUCCEEDed when the pass was not completed. In the case where an opposing player actually knocked the ball away from the would-be receiver, our system would make a hypothesis of a local competitive interaction in which the would-be receiver FAILed, because he did not catch

the ball, and the opposing player SUCCEEDed because he prevented the former player from achieving his goal. The attribution of higher level goals requires the introduction of more global information. We deal with this problem in Section IV.3.5.

A stronger implication than (2) can be drawn about the SUCCEED-FAIL relationship of the players' competing goals. Namely, it must be the case that for any competitive interaction both clauses of the above disjunction must be possible, i.e., either player in the competitive-interaction must in principle be able to win (succeed). Indeed, it would be a strange "game" if only one team could win! In the pitcher-batter example of Section IV.2, we saw the batter (PLAYER2) SUCCEED while the pitcher (PLAYER1) FAILs. Following the above claim, the pitcher must be able to SUCCEED and the batter FAIL in order to rightfully call their interaction competitive. From actions in baseball (Appendix), we in fact know that it is often the case that the pitcher THROWs a BALL and the batter does NOT HIT the BALL.[13] Moreover, in the example of Section IV.2, we saw how the system reasoned hypothetically to verify the possibility of the alternative outcome.

Assume that the system is told about various different kinds of competitive causal relations often found in games (e.g., PHYSICAL COMPETITION). Then, for each type of competitive relation there are two possible outcomes: FAIL(PLAYER1)/SUCCEED(PLAYER2) and SUCCEED(PLAYER1)/FAIL(PLAYER2). (The former is abbreviated F/S, while the latter is abbreviated S/F.) Correspondingly, for each type of competitive relation there are two CLSs; a

version-1-CLS recognizes the specific kind of competitive relation and recognizes the FAIL/SUCCEED outcome structure, while a version-2-CLS recognizes the same kind of relation but with a SUCCEED/FAIL outcome structure. Different tests need to be made in order to discriminate between an S/F or an F/S outcome. Thus, two distinct rules are required, rather than one, for each type of relationship.

We are now in an interesting position. We can define the goals of the players independently of the particular competitive relationships about which the system knows. In order to achieve the FAIL/SUCCEED outcome structure, the goals of the players must be as follows: the GOAL-OF(PLAYER1) must have been to PREVENT PLAYER2's action, and since PLAYER2 did execute that action in the particular interaction under investigation, PLAYER1 FAILed with his goal. Similarly, PLAYER2 must have wanted to execute his action, and thus he SUCCEEDed in the particular interaction. The Causal-Link Schemas for the <u>FAIL/SUCCEED</u> version of all the competitive relationships have the following form:

$$\text{TYPE-OF COMPETITIVE-INTERACTION}_{\text{VER1}}(\text{ACT-OF(PLAYER1)},\text{ACT-OF(PLAYER2)}) \rightarrow$$
$$\text{GOAL-OF(PLAYER1)} = (\text{WANT PREVENT ACT-OF(PLAYER2)}) \wedge$$
$$\text{GOAL-OF(PLAYER2)} = (\text{WANT EXECUTE ACT-OF(PLAYER2)})$$

where PLAYER1 and PLAYER2 are on opposing teams and the ACT-OF(PLAYER1) can be said to have "enabled" the ACT-OF(PLAYER2).

The version-1-CLS for the competitive relation PHYSICAL-COMPETITION in the example of Section IV.2 made goal hypotheses of the above sort.

A similar analysis can be made for that version of a CLS which recognizes the SUCCEED/FAIL outcome structure. In order to achieve that outcome, PLAYER2 must <u>not</u> have wanted to execute the action he did in

fact execute, thus he FAILed in the interaction under investigation. On the other hand, PLAYER1 must have wanted PLAYER2 to execute that action, and since PLAYER1's action "enabled" in some sense PLAYER2's undesired action, PLAYER1 SUCCEEDed in the interaction. The CLSs which recognize the SUCCEED/FAIL outcome have the following form:

$$\text{COMPETITIVE-INTERACTION}_{VER2}(\text{ACT-OF(PLAYER1)},\text{ACT-OF(PLAYER2)}) \rightarrow$$
$$\text{GOAL-OF(PLAYER1)} = (\text{WANT ENABLE ACT-OF(PLAYER2)}) \wedge$$
$$\text{GOAL-OF(PLAYER2)} = (\text{NOT WANT EXECUTE ACT-OF(PLAYER2)})$$

where PLAYER1 and PLAYER2 are on opposing teams, and where the ACT-OF(PLAYER1) can be considered to have "enabled" the ACT-OF(PLAYER2).

We shall see an example of this type of CLS in the next section.

This discussion was intended to provide a general framework for the rules which are to follow. We shall (1) present three of the four types of competitive interactions about which the system is aware, (2) discuss the criteria used to establish when a particular relationship exists, and (3) discuss how the CLSs distinguish between a FAIL/SUCCEED outcome and a SUCCEED/FAIL outcome. This will be followed by a discussion of the cooperative interactions known to the system.

IV.3.2.1 PHYSICAL-COMPETITION

One could say that the basic competitive relationship in an action game is "PHYSICAL-COMPETITION". Recall from the example in Section IV.2 that this type of competitive relation was hypothesized for the batter-pitcher interaction of Figure IV.1. More particularly, it was decided that the outcome structure of this interaction was FAIL(PLAYER1)/SUCCEED(PLAYER2), where the pitcher A1 was PLAYER1 and the batter B1

was PLAYER2. This decision was based on essentially the following points: (1) the inference that both the pitcher's THROW and the batter's SWINGHIT could be considered difficult-acts, (2) the inference that the pitcher's THROW enabled the batter's SWINGHIT, (3) the assumption that difficult-acts are executed intentionally, (4) the assumption that execution (prevention) of such acts are the goals of the players, and (5) the assumption that one player does not want to enable the intended act of an opposing player. Thus, the version-1-CLS for PHYSICAL-COMPETITION hypothesized that the goal structure was F/S, with the pitcher FAILing to PREVENT his opponent's difficult-act, SWINGHIT, while the batter SUCCEEDed with his intended goal by executing the SWINGHIT.

The snapshots in Figure IV.3 depict an instance where the outcome goal structure is S/F, i.e., the pitcher A1 SUCCEEDs while the batter B1 FAILs. The CLS for the S/F version of the PHYSICAL-COMPETITION relation determined that the action SWING a bat and MISS the BALL (not HIT the BALL) was not a difficult-act. Thus, the inference can be made that B1 did not intend to execute the action and therefore FAILed, while his opponent A1 wanted to enable B1 to execute SWINGMISS and therefore he SUCCEEDed.

From the inference of an F/S competitive relationship, in the pitcher-batter example, the system could predict the S/F version; i.e., from [(PITCHER THROW)-(BATTER SWINGHIT)] we can predict [(PITCHER THROW)-(BATTER NOT(SWINGHIT))], where SWINGMISS is a type of NOT(SWINGHIT). Similarly, the S/F version could be used to predict the F/S version.

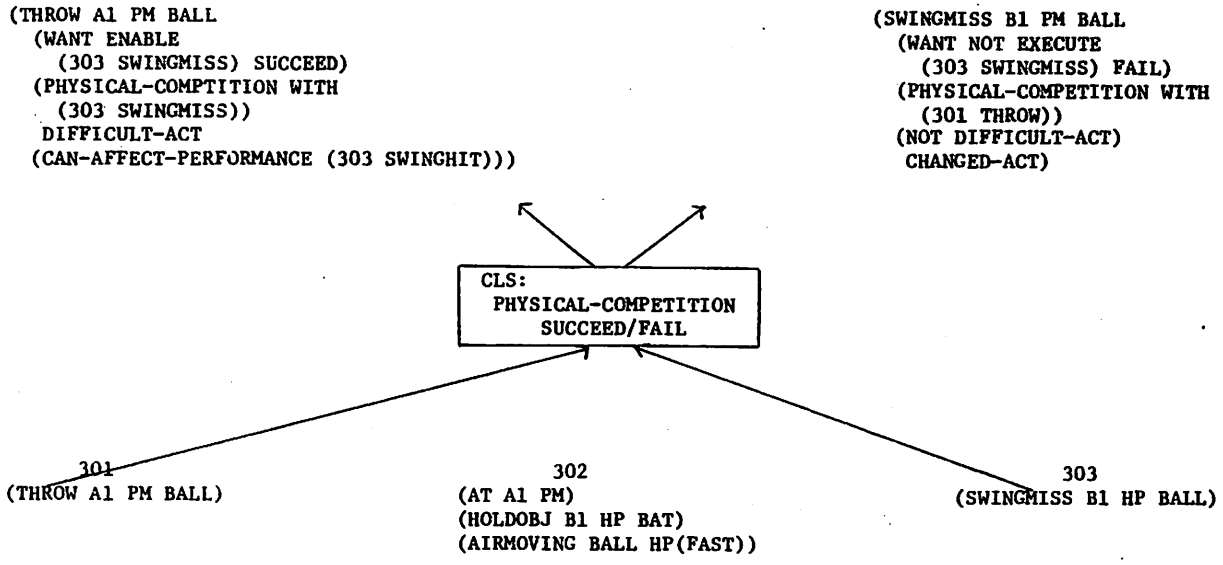


Figure IV.3 An Example of PHYSICAL-COMPETITION with a SUCCEED/FAIL Outcome

While the competitive interaction depicted in Figure IV.1 resulted in a FAIL/SUCCEED outcome structure, in the above interaction we see an example of the complimentary outcome structure, SUCCEED/FAIL. It was inferred that the act which B1 executed was probably not his intended act. Since A1's act enabled B1's FAILure, we can hypothesize that A1 SUCCEEDed while B1 FAILed with their respective goals. The pattern descriptions are augmented to reflect this hypothesis.

These predictions follow from the basic outcome structure of a competitive interaction; both players in a competitive interaction must in principle be able to SUCCEED. In Chapter VII we shall discuss how these predictions are used to verify the goal and competitive relationship hypotheses.

IV.3.2.2 ORDER-OF-OCCURRENCE-COMPETITION

There are other aspects of spatio-temporal actions which can serve to link two actions together. One that is natural for action games is the relative order of occurrence of two actions in time. The completion of an action before or after another action may define a relationship. In fact, such relationships are commonplace in games such as baseball, basketball, football, tennis, etc. The snapshots in Figure IV.4 depict the final events of an infield single. We see player B1 arriving at a location called FIRSTBASE before the opposing player A3 at that same location CATCHs the BALL. The reason B1 is permitted to remain ON FIRSTBASE is that he was able to perform that action before his opponent executed the CATCH action. Thus, the relation ORDER-OF-OCCURRENCE in time is used to define another type of competitive interaction.

In order to hypothesize that an ORDER-OF-OCCURRENCE relationship might explain why PLAYER2[14] changed from executing one action to executing ACT2, the system must find an action, ACT1, performed by opposing player, PLAYER1, which occurs immediately before or shortly after ACT2, the action in question. For example, assume that in Figure IV.4 the system is trying to explain why B1 changed from RUNning to not-RUNning

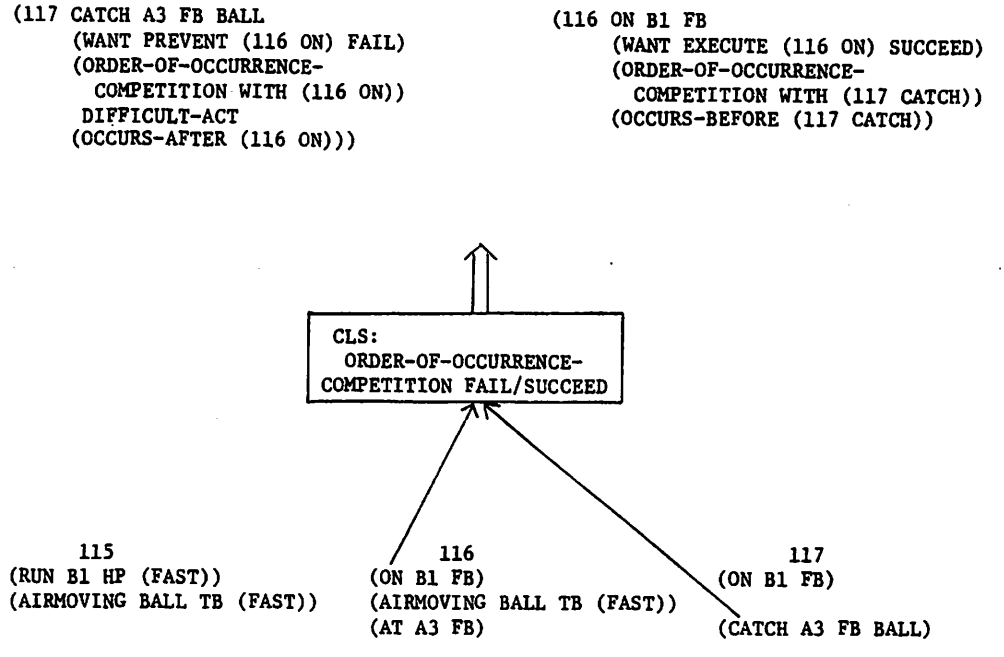


Figure IV.4 An Example of ORDER-OF-OCCURRENCE COMPETITION with a FAIL/SUCCEED Outcome

The CLS triggered by the above action sequence hypothesized that the reason B1 was permitted to remain ON FIRSTBASE was because he performed that act before A3 caught the BALL at FIRSTBASE. While we depict here only the appropriately augmented pattern descriptions, the system of course produces a production rule representation of the same information.

in terms of this timing relationship. A3's act CATCH would satisfy this first condition of this relation since it occurred shortly after B1's act ON.

Next, the ORDER-OF-OCCURRENCE relationship assumes that PLAYER2 wanted to execute ACT2 and that he was allowed to execute that act because he executed it before an opposing player, PLAYER1, executed ACT1. For example, since B1 executed the act ON before A3 caught the BALL in the snapshots of Figure IV.4, we can infer that B1 intended to execute that act. We can also infer from our assumption that players engaged in a competitive interaction do not want to enable the opposition to perform actions which the opposition intends to perform, that A3 could not have intended (wanted) that B1 execute ON FIRSTBASE. Thus, we can hypothesize that A3 FAILed by not PREVENTing B1 from getting ON FIRSTBASE, while B1 SUCCEEDed by doing just that. This particular interaction exhibits the F/S outcome structure.[15]

Just as in the case of PHYSICAL-COMPETITION, the system must verify that the alternative (complimentary) outcome was possible. Using the procedure CAN-AFFECT-PERFORMANCE, the system determines that in fact A3 could have caught the BALL SOONER, if his teammate, A5, had THROWn it FASTER. This would imply that A3 would have caught the BALL before B1 executed ON FIRSTBASE, and thus presumably B1's action would not have been permitted. Similarly, B1 could have reached FIRSTBASE SOONER, if he had RUN FASTER. Therefore, the necessary variability in this competitive interaction was present. Figure IV.4 depicts the augmented pattern

description produced by the successful triggering of the ORDER-OF-OCCURRENCE CLS in this example; note that a corresponding production rule representation is also produced (not illustrated).

As we said earlier, we can predict the S/F structure from the F/S structure, and vice versa. Figure IV.5 depicts the snapshots of events which illustrate the S/F structure; in baseball these events would be called an infield groundout. Notice that the batter B1 changed his actions immediately after the opposing player A3 caught the BALL. In other words, when A3 did CATCH the BALL earlier, B1 was not permitted to execute ON FIRSTBASE.

IV.3.2.3 STATEOF-DISTINGUISHED-OBJECT COMPETITION

While the preceding competitive CLSs related one player's change in actions to an opponent's action, the CLS STATEOF-DISTINGUISHED-OBJECT relates one player's change in actions to the conjunction of an opponent's action and the history of the distinguished object-- the BALL. This CLS is an attempt to introduce 'state' information into the analysis. However, the STATEOF-DISTINGUISHED-OBJECT relationship is a "weak" one in that it almost always can be used to explain a change in a player's actions. The basic requirement is that the player in question must have been involved with the BALL (THROWn it or HIT it) at some earlier point.

Consider the snapshots in Figure IV.6 taken from an outfield single. The version-1-CLS (F/S) for this relationship hypothesizes that B4 was allowed to execute ON FIRSTBASE because the BALL touched the GROUND before it was caught by an opponent, A8. Thus, B4 SUCCEEDed with his goal of

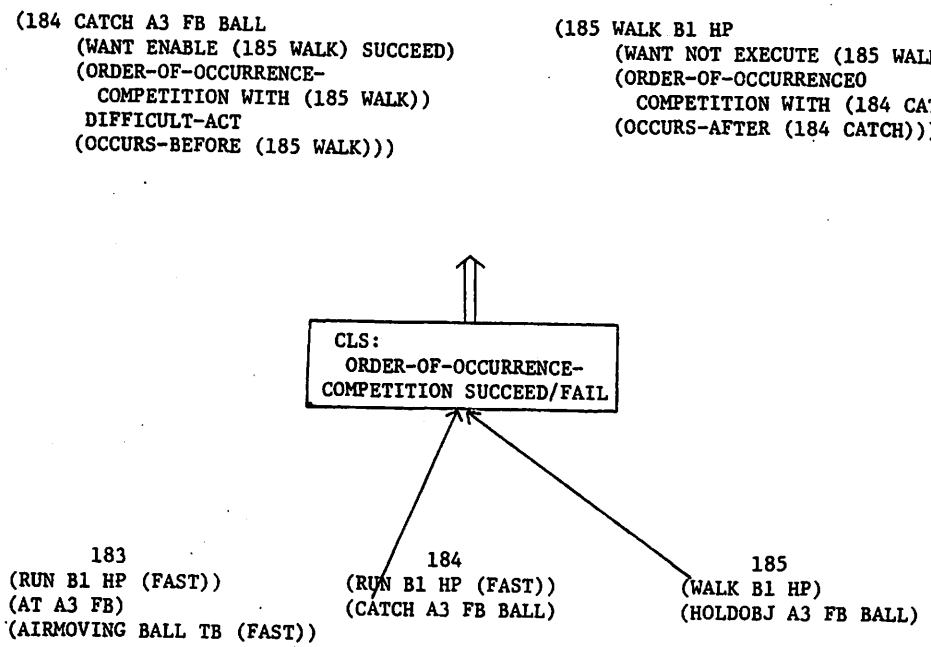


Figure IV.5 An Example of ORDER-OF-OCCURRENCE COMPETITION with a SUCCEED/FAIL Outcome

In the above action sequence we see player A3 CATCHing the BALL before B1 reaches FIRSTBASE, and thus B1 is not permitted to execute ON FIRSTBASE; as opposed to the example in Figure IV.4, this time A3 SUCCEEDed and B1 FAILed with their respective goal. Notice that from the competitive interaction of Figure IV.4, it can be predicted that the above interaction should take place.

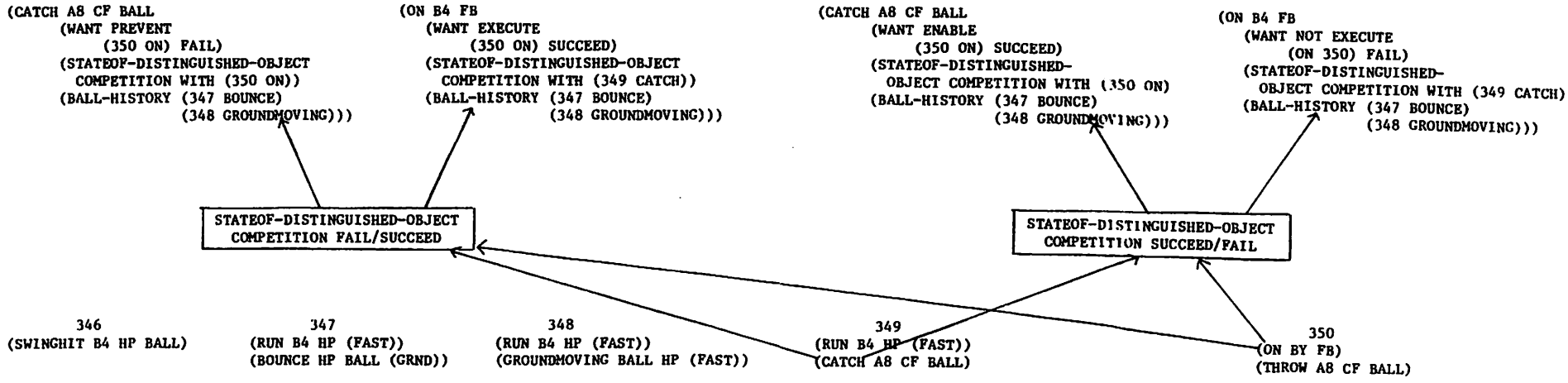


Figure IV.6 An Example of STATEOF-DISTINGUISHED-OBJECT-COMPETITION

Both the S/F and F/S CLSs for this relationship are triggered by the action sequence depicted above.

getting ON FIRSTBASE, while A8 FAILed with his by not CATCHing the BALL before it hit the GROUND and thereby enabling B4's action. As is often the case with this relationship, the conditions for triggering the version-2-CLS (S/F) are also satisfied by the same situation which triggers the version-1-CLS. Thus, in the situation depicted in Figure IV.6, the version-2-CLS hypothesizes that A8's CATCH in conjunction with the history of the BALL (BOUNCing on the GROUND after being HIT, and then MOVing on the GROUND) forced B4 to execute ON FIRSTBASE. In other words, this hypothesis suggests that B4 did not want to HIT the BALL on the GROUND and, as a result by doing so, was not permitted to execute some other action; thus, getting ON FIRSTBASE was not considered to be a desirable act for B4.

Clearly, the hypothesis of the version-2-CLS is wrong. However, the goal hypothesis of the version-1-CLS is correct, but for the wrong reason; the BALL's touching the ground before an opponent catches it is a necessary but not a sufficient condition for B4 to be allowed to execute ON FIRSTBASE. The flaw is the system's single-mindedness in attempting to interpret a player's change in his action as a result of a competitive interaction with an opposing player. No particular A-team player was at fault for B4's achieving his goal of reaching FIRSTBASE; nonetheless, B4 achieved a competitive goal. What the system needs are CLSs which could hypothesize an achievement (or failure) of a competitive goal without having to find a specific player to "blame". Finally, in Chapter VII we discuss how the verification of other hypotheses facilitates the resolution of similar conflicts in interpretation.

IV.3.3 Local Cooperative Interactions

In the previous section we saw how the system tried to explain a change in a player's action sequence by establishing a link to an opponent's action. Another way to explain why a player executed a change in his action sequence is to appeal to a COOPERATIVE-INTERACTION which "enabled" the change. In order to hypothesize a COOPERATIVE-INTERACTION, the system tries to establish a link to a previous action by the same player or by a teammate of the player. Consider the snapshots in the bottom of Figure IV.7 taken from the end of an infield single episode. The system will hypothesize a cooperative relationship between A3's CATCH and his teammate A6's THROW. (Ignore for a moment A3's AT which immediately precedes his CATCH.)

Just as in the case of competitive interactions, we can define a goal structure for cooperative interactions independent of the particular type of cooperative relationship.

COOPERATIVE-INTERACTION(ACT-OF(PLAYER1),ACT-OF(PLAYER2))→
[GOAL-OF(PLAYER1) = WANT ENABLE ACT-OF(PLAYER2)] ∧
[GOAL-OF(PLAYER2) = WANT EXECUTE ACT-OF(PLAYER1)]

where PLAYER1 and PLAYER2 are either the same player of players on the same team, and PLAYER1's action "enabled" PLAYER2's action.

For example, the GOAL-OF A6's THROW was to enable A3's CATCH, and A3's goal was to CATCH the BALL. At the level of <u>local</u> cooperative interaction, the outcome (SUCCEED-FAIL) structure is trivial; we simply label both parties to the cooperative interaction as SUCCEEDing.[16] Since there is no alternation of SUCCEED-FAIL outcomes, the system does not need two

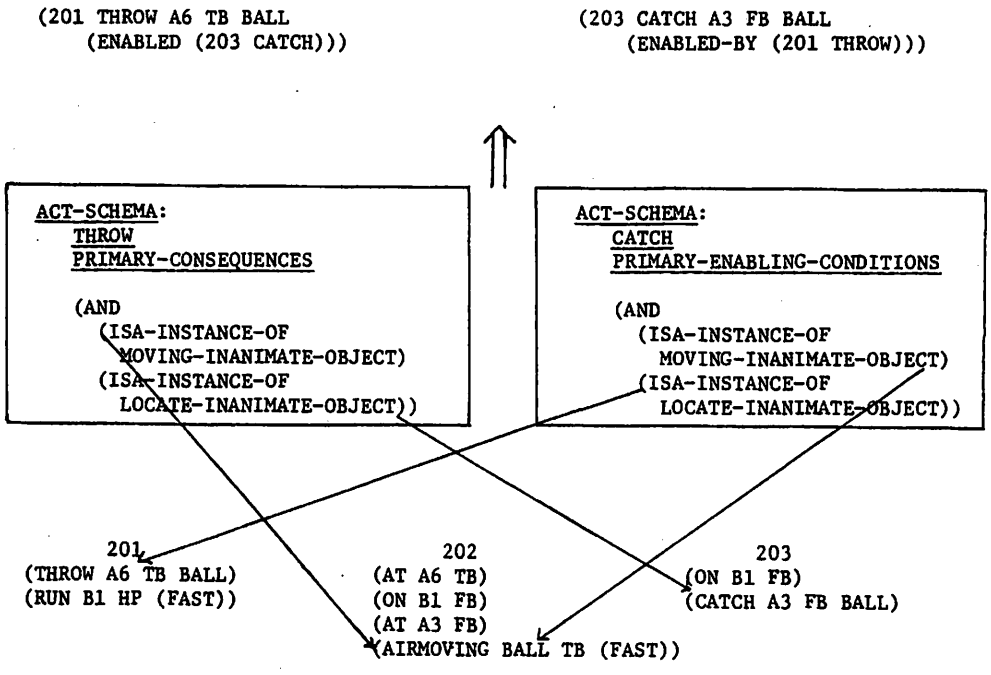


Figure IV.7 A Physical Relationship Between Members of the Same Team

In the above action sequence we see from the abbreviated act-schemas that the PRIMARY-CONSEQUENCES of A6's THROW satisfied the PRIMARY-ENABLING-CONDITIONS for A3's subsequent CATCH. The pattern descriptions are augmented to reflect this inference.

versions (two CLSs) of each cooperative relationship. Furthermore, there are only two defined cooperative relationships: PHYSICAL-COOPERATION and ORDER-OF-OCCURRENCE-COOPERATION.

PHYSICAL-COOPERATION hypothesizes a cooperative interaction between two players on the same team on the basis of physical-enabling conditions. In particular, in order to say that PLAYER1's act, ACT1, COOPERATIVELY-ENABLED PLAYER2's act, ACT2, the PRIMARY-PHYSICAL-ENABLING-CONDITIONS (preconditions) of PLAYER2's act must be satisfied by the PRIMARY-CONSEQUENCES of PLAYER1's act. For example, in the top half of Figure IV.7 we see a partial description of the act-schemas for THROW and CATCH. From the act-schemas we can see that the PRIMARY-ENABLING-CONDITIONS for CATCH are satisfied by the PRIMARY-CONSEQUENCES of THROW. The attempt here is to capture some sense of intentionality, when A6 threw the BALL he intended to enable A3 to CATCH it. Notice that A3's AT FIRSTBASE action (snapshot 202) is a necessary but incidental precondition for his subsequent CATCH at that location; the PRIMARY-CONSEQUENCES of AT satisfy the SECONDARY-ENABLING-CONDITIONS of CATCH. Recall that the SECONDARY-ENABLING-CONDITIONS identify those aspects of an action which are not directly related to the main function (PRIMARY-CONSEQUENCES) of the act. Figure IV.8 illustrates the features added to the pattern descriptions of the snapshots due to the hypothesis of a PHYSICAL-COOPERATIVE-INTERACTION. (As always, there is a corresponding specific CLS generated by the general CLS making the inference.)

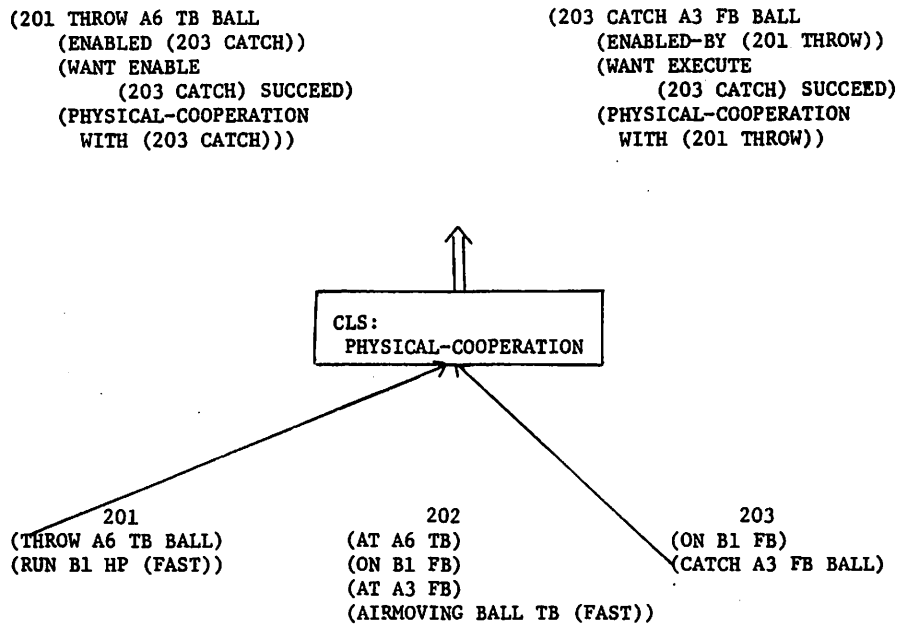


Figure IV.8 An Example of PHYSICAL-COOPERATION

The inference by the act-schemas that a direct physical enablement relationship exists between A6's THROW and A3's subsequent CATCH, plus the observation that A6 and A3 are on the same team, are interpreted by the CLS PHYSICAL-COOPERATION as indicating a cooperative interaction. The firing of a cooperative CLS also results in the generation of appropriately augmented pattern descriptions and a new CLS.

While PHYSICAL-COOPERATION requires a physical link with a previous action, ORDER-OF-OCCURRENCE-COOPERATION requires a logical link with a previous action. Namely, PLAYER2 was allowed to execute his act, ACT2, because he executed that act after a teammate, PLAYER1, executed an act, ACT1. This type of cooperative relationship is similar in spirit to its competitive counterpart, ORDER-OF-OCCURRENCE-COMPETITION. In Figure IV.9 we see an example of ORDER-OF-OCCURRENCE-COOPERATION between B1's SWINGHIT and his subsequent RUN. In this interaction, the PRIMARY-ENABLING-CONDITIONS of RUN are satisfied by the SECONDARY-CONSEQUENCES of the preceding SWINGHIT (see Figure III.6). Thus, only in an indirect way does the SWINGHIT enable the RUN of Figure IV.9. We interpret this "INDIRECT-PHYSICAL-ENABLEMENT" as one of logical enablement (see Figure IV.9); this type of cooperative interaction emphasizes a logical relationship of precedence in time -- in order to be permitted to RUN, a player must first HIT the BALL.

In Figure IV.10 we also see that ORDER-OF-OCCURRENCE-COOPERATION was hypothesized to explain why player B2 was allowed to change from standing ON FIRSTBASE to RUNning from FIRSTBASE; B2 was allowed to do so because B1 previously HIT the BALL. Actually, this interpretation is not quite correct; the rules of baseball would permit B2 to RUN from FIRSTBASE at any time (called "stealing a base"). The current set of CLSs do not include one which hypothesizes the state as the enabling condition (see Section IV.3.4). However, the above interpretation is certainly a reasonable one for situations where players (teammates) are seen RUNning after the BALL has been HIT.

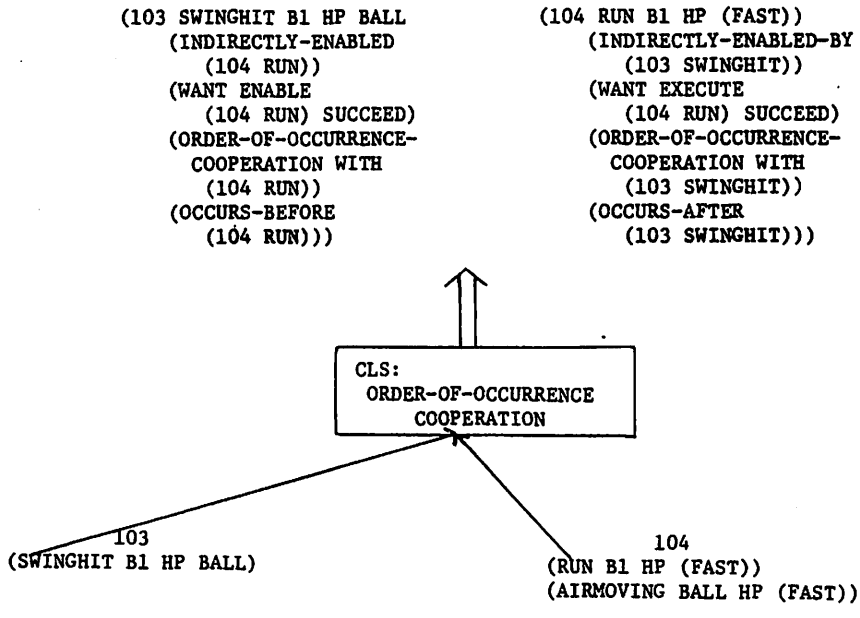


Figure IV.9 An Example of ORDER-OF-OCCURRENCE-COOPERATION

Analogous to the notion of ORDER-OF-OCCURRENCE-COMPETITION is the notion of ORDER-OF-OCCURRENCE-COOPERATION. In the above action sequence, it is hypothesized that the reason B1 was allowed to start RUNning towards FIRSTBASE was due to his previous act of HITting the BALL.

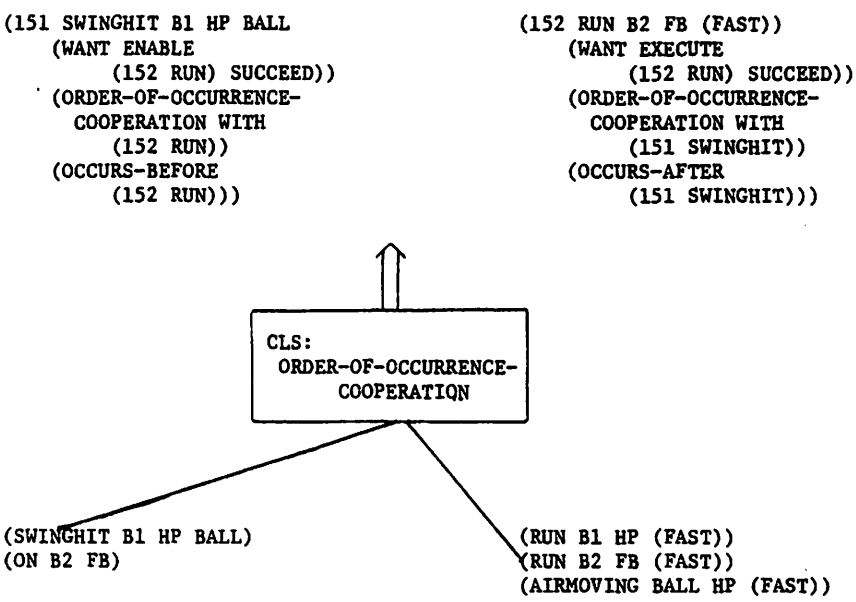


Figure IV.10 ORDER-OF-OCCURRENCE-COOPERATION Between Two Different Players on the Same Team

A timing relationship may exist between two different players on the same team. In the above action sequence, we see that B1's HIT apparently enabled B2's subsequent RUN. This hypothesis was suggested since the system saw B2 start RUNning after B1 HIT the BALL.

The distinction between a logical cooperation, such as ORDER-OF-OCCURRENCE, and PHYSICAL-COOPERATION is an attempt to capture some additional structure in cooperative relations. However, the system does not make any important use of the distinction. If a deeper analysis between cooperative and competitive relations was performed, this distinction would probably be important.

IV.3.4 The Basis of a Relationship: Generating New General CLSs

Now that we've seen some examples of competitive and cooperative relationships, we can better appreciate the following analysis of their "origins". Consider the pattern descriptions of the pitcher-batter interaction depicted in Figure IV.11. Forget for a moment that we said that CLSs and act-schemas in fact add features such as DIFFICULT-ACT, PHYSICALLY-ENABLES, etc. That is, assume that such features are already present in the pattern descriptions. Essentially, the PHYSICAL-COMPETITION CLS, which expresses the general concept of a physical competitive relationship in games, selects from a set of features, such as the ones in Figure IV.11, a subset of features (DIFFICULT-ACT, SWINGHIT, PHYSICALLY-ENABLES, etc) and identifies that subset as signifying an important relationship. The CLS which recognizes a timing relationship, ORDER-OF-OCCURRENCE, does likewise; however the subset it chooses is different (though not necessarily disjoint) from the subset which represents the PHYSICAL-COMPETITION relationship. In other words, a competitive (or cooperative) relationship specifies that only certain aspects (features) of the situation are the relevant or determining factors in the relationship.

ABSOLUTE TIME	ACTION	PLAYER	LOCATION	DEGREE-OF DIFFICULTY	PHYSICAL ENABLEMENT	RELATIVE TIME	MISCELLANEOUS
(101	THROW	A1	PM	DIFFICULT	ENABLES (SWINGHIT 103)	OCCURS BEFORE (103 SWINGHIT)	CAN-EFFECT-PERFORMANCE (103 SWINGHIT))
(103	SWINGHIT	B1	HP	DIFFICULT	ENABLED-BY (THROW 101)	OCCURS AFTER (101 THROW)	CHANGED-ACT)

Figure IV.11 Pattern Descriptions for the Pitcher-Batter Competitive Interaction

Essentially, any subset of features could specify an important relationship. For example, we do not have a CLS which recognizes that location is a determining aspect of a situation. As such, the system misses the fact that a player can only HIT the BALL from HOMEPLATE.[17] A CLS could be generated which includes this feature. However, any arbitrary subset of features would probably not have a meaningful interpretation in games. Nonetheless, new relationships -- new causal-link schemas -- could be defined by carefully picking interesting subsets of features.

The set of features from which new CLSs may draw need not be limited to ones which reference only a pair of pattern descriptions (actions). While the CLSs in the current system hypothesize relationships only between two players, CLSs which reference the actions of a number of players may be needed. As we saw in Section IV.3.2.3, CLSs are also needed which could hypothesize an achievement (or failure) of a competitive goal without having to find a specific player to "blame". That is, relationships between a player's action and the state of the world are also important.

IV.3.5 The Problem of Composing Local Goals into Plans[18]

Action games such as baseball are a type of social situation in which goal-directed entities interact with each other. Since the goals implicit in social situations seem to be structured hierarchically, it follows that the goals in action games might also be similarly structured. While there has been some work at attempting to formalize the goal structure of human activity in various contexts(e.g., RUM75, SCHA77, SCHM77b, SCHM76b), there is currently no description of the goal structure for social interactions in general.

It has been argued that action games[19] are stylized and miniaturized versions of the general social interactions evident in a society. One could hope then, that for such apparently constrained social activity as action games, a "grammar" could be abstracted which would describe the goal structure of games. With such a "grammar" one could describe the final distinguished goal which determines "winning" and "losing" as some composition of goals implicit in the local interactions of the players -- for all action games. Given the diversity of action games, one wonders if, even for this subset of human activity, a general description is possible -- or doable. While we have clearly not solved this enormous problem, we have provided our learning system with a crude description of two levels in the goal hierarchy. The previous few sections described our conception of the "lowest" level of goals, that of local competitive and cooperative interactions. In this section we will proceed to the "next" level of goals by describing in a rough way the goal structure of a competitive "episode".

IV.3.5.1 The Structure of a Competitive Episode

We view the elucidation of the hierarchical goal structure of games as the identification of levels of pattern descriptions (see Figure II.1). We have argued that the levels of pattern descriptions which we have chosen up to this point reflect a natural decomposition of the structure of games. For example, the local competitive and cooperative interactions of players seem to be the lowest level of goals in a game.[20] The next level of goals seems to coincide with the structure of "episodes", high activity sequences surrounded by low activity sequences. Most physical games have such a

"unit" of description. In tennis, an episode would be the volleying for a point; in football an episode would be the passing or running in a "down".

One important property of a competitive episode which we have identified is that it must contain at least one competitive interaction,

```
COMPETITIVE-EPISODE(ACTS-OF(TEAM-A),ACTS-OF(TEAM-B))→
∃(PLAYER1,PLAYER2)(COMPETITIVE-INTERACTION(ACT-OF
(PLAYER1),ACT-OF(PLAYER2))).
```

Obviously, this is a weak constraint. A stronger constraint would be to specify a syntax for sequences of interactions. We attempted to define such a grammar but were forced to abandon that effort. Either the grammars which were produced were specific to baseball or they were as general as the above simple constraint. The problem seems to lie in the nature of games -- they seem relatively arbitrary, almost any type of interaction could follow any other. A deeper analysis of games might turn up "universals" of games in the same sense as there seems to be universals in language [BAC74]. Following the linguistic analogy, the universals in games would narrow down the possible space of games by eliminating certain types of constructions. They would specify that only certain types of interactions could follow (or precede) other types. We are not aware of any researchers who claim to have discovered such universals in games.

We briefly discussed the other major structural property of competitive episodes earlier. Namely, competitive episodes have goal outcomes, just as local competitive interactions have goal outcomes. In particular,

the goals of a competitive episode are the goals of the final competitive interactions in the episode,[21]

```
GOALS-OF(COMPETITIVE-EPISODE(ACTS-OF(TEAM-A),ACTS-OF(TEAM-B))) =
GOALS-OF(FINAL-COMPETITIVE-INTERACTION(ACT-OF(PLAYER1),ACT-
OF(PLAYER2)))
```

Intuitively, the final goal of a plan summarizes all the activity which went into making up the plan. In our characterization, an episode is a first level plan (Figure II.1, Level 6). For example, we can summarize all the competitive and cooperative interactions which constitute an infield single (Figure IV.12) by saying that the goal of player B1 was to get ON FIRSTBASE, while the goal of the opposing player A3 was to prevent B1's ON. In fact Schmidt [SCHM76b] argues that a "legal" summary for non-social interactions such as those in the story in Table V.1, need be nothing more than simply stating the final goal of the plan.

There seems to be a natural relationship between the final competitive goals in an episode and the local goals. We can say that the local goals were achieved in order to enable the final goal. Consider the pattern descriptions for the pitcher-batter interactions in Figure IV.12. Notice that the goal of the batter B1 was hypothesized to be that of performing the action SWINGHIT. A reasonable question would be "why is that a desirable goal?" A reasonable answer would make reference to the batter's final goal, namely, B1 wanted to execute SWINGHIT so as to enable him to execute ON FIRSTBASE, the final competitive goal of the episode. Similarly, the next logical question would be "why is getting ON FIRSTBASE a desirable goal?" An answer to this question might make reference

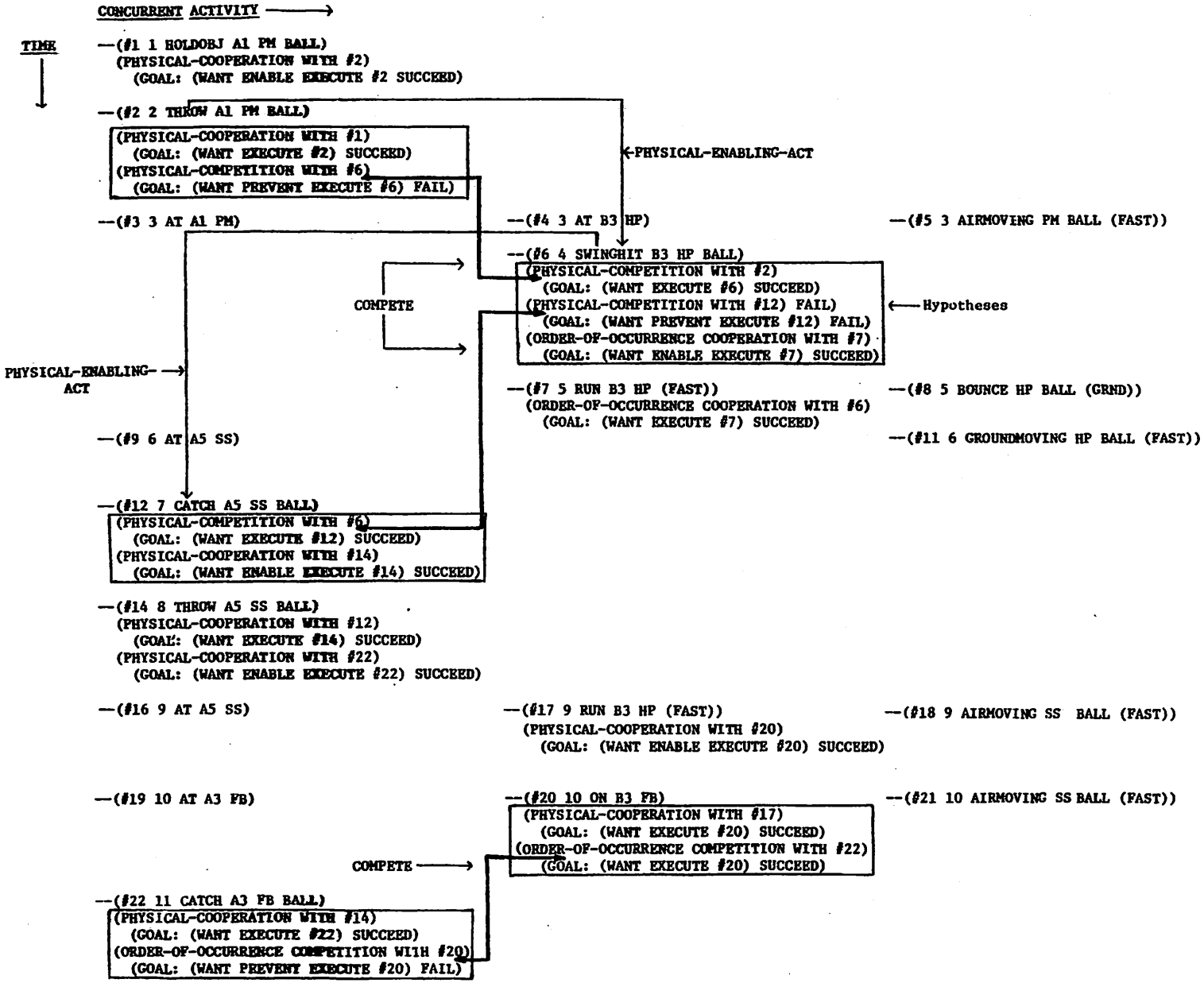


Figure IV.12 An Infield Single Episode After Interpretation

to the final distinguished goal in baseball, that of traversing the bases and scoring a run.

Consider again local cooperative interactions, e.g., the one between the shortstop A5 and the first baseman A3 depicted in Figure IV.12. The trivial assignment of SUCCEED to both actions does not take into consideration the subsequent competitive interaction which was enabled by the local cooperative interaction. A more accurate assessment of the cooperative goals of the involved players would make reference to the outcome of the competitive interaction. In the above case we might want to assign FAILure to the goal of the cooperative interaction of A5 and A3 since it resulted in a FAILure in the competitive interaction of A3 and B1 which it enabled.

While it seems clear that there is some close relationship between the local goals and the final competitive goals of an episode, it is another question to rigorously define that relationship. For example as we noted above, A5's THROW bears some relation to the final competitive goals of the episode depicted in Figure IV.12. However, it is not clear what the pitcher A1's relationship is to the final competitive interaction; A1's THROW enabled B1's SWINGHIT which enabled the final competitive interaction. However A1 FAILed with his goal to PREVENT B1's SWINGHIT. If A1 had SUCCEEDed with this goal, then this would have prevented the possible subsequent SUCCESS by A3 at FIRSTBASE. In general, how is responsibility for the final goals distributed to the local goals? Is there some additional internal structure to an episode, besides the

highlighted final competitive interaction? A general answer to this question awaits a more comprehensive description of the relationship of subgoals and goals in social interactions. As a first step in that direction we shall now attempt to describe what the internal structure of an infield single seems to us to be.

IV.3.4.2 A Proposed Description of the Internal Structure of a Competitive Episode[22,23]

We suggest that local competitive interactions impart a hierarchical structure on a competitive episode; such interactions define subgoals which are performed in order to achieve the final competitive goal of the episode. The diagramming of an infield single in Figure IV.13 reflects this observation; the double arrows which lead from a competitive interaction into a triangular shaped box indicate levels of structure internal to an episode. Two benefits derive from this representation. First, placing subgoals above actions captures the notion that different sequences of actions can achieve the same subgoal. For example in baseball, a "walk" (in its technical sense) is an act sequence different from THROW-SWINGHIT (Figure IV.13) which accomplishes the same subgoal as the THROW-SWINGHIT, i.e., the batter is permitted to RUN to FIRSTBASE. Second, a bound can be placed on how far responsibility is "passed" to the cooperative interactions which enable competitive interactions. Namely, pass back the appropriate outcome (SUCCEED or FAIL) to cooperative interactions, but do not integrate cooperative interactions over subgoal boundaries. For example, this rule would integrate A5's THROW with the final competitive interaction between A3 and B1 (Figure IV.12), but would not blame

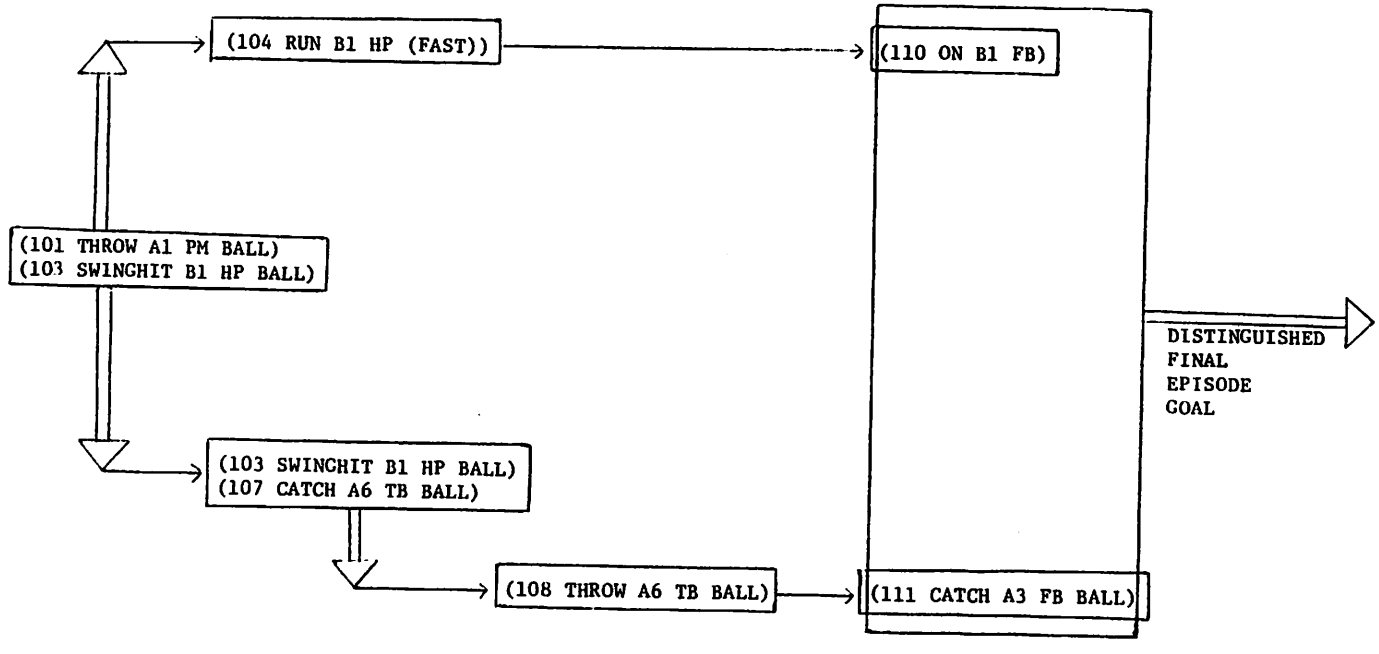


Figure IV.13 Proposed Internal Structure of an Episode

The episode illustrated in this diagram is that of an infield single (Figure IV.12). Competitive interactions are indicated by double arrow leading into a triangular box; such interactions signify major subgoals in the episode. The final goal of the episode is the one for which the subgoals were achieved.

the pitcher A1's THROW for team-A's FAILure in the final competitive interaction.

The ideal here is that competitive interactions are SUCCEED/FAIL points; one team can FAIL at one point but SUCCEED at another. Thus, if A1 had SUCCEEDed with his goal of PREVENTing B1 from HITting the BALL, then the episode would have ended there. Since A1 FAILed at this juncture, the rules of baseball give team-A another chance at PREVENTing B1 from SUCCEEDing with his episode goal, namely, have A3 CATCH the BALL before B1 arrives at FIRSTBASE. In other words, the success (failure) of a sequence of subgoals does not necessarily imply success (failure) of the final goal.

While we have not explicitly mentioned it previously, it is possible -- in fact likely -- that multiple CLSs will trigger and hypothesize goals for a single action of a player. A complete theory of the internal structure of a competitive episode would integrate multiple goals hypothesized for a single action. For example, in Figures IV.12 and IV.13 we see that 3 CLSs have been triggered and have hypothesized goals for B1's SWINGHIT. One competitive CLS suggests that B1 SUCCEEDed with respect to one goal, but another competitive CLS suggests that he FAILed with respect to another goal. However, it seems that B1 SUCCEEDS with respect to his final competitive goal, getting ON FIRSTBASE. Some "relabelling" of SUCCESSes and FAILures seems to be in order in this case. We suggest that a rule such as the following one captures our intuition for that relabelling:

If a person FAILs at a subgoal but SUCCEEDs (or his team SUCCEEDs) at a final goal, a higher order relabelling should occur and the FAILed subgoal marked as unimportant.

IV.3.6 Suggestions for Extending the Knowledge Base: Acquiring an Understanding of Higher Level Goals[24]

Recall from an earlier discussion (Section IV.3.5.1), that the goal of the batter in an infield single (Figure IV.12) was hypothesized to be performing the act ON FIRSTBASE. We also raised the logical question "why is that a desirable goal?" An answer to that question would require a reference to a higher level goal of which this action was a subgoal. In this section, we would like to briefly speculate on the knowledge that might be needed in order to resolve the above question: How could our system discover higher level goals? Following the previous line of reasoning, the system would have to have general rules which recognized when such an event had taken place; the rules would interpret some event as an instance of a higher level goal. Also, the system would have to actually observe such an event; it cannot predict what the goal would be, since the rule which it has is expressed only in terms of features (properties) of an event. An example of such a rule might be "when the scoreboard marker which counts 'distinguished final goals' changes, notes the actions which occurred prior to the change." Essentially, correlate a change in a marker with an action(s) or goal(s). However, which marker is the important one? There are at least 10 markers in baseball: runs-for-the-A-team, runs-for-the-B-team, hits-for-the-A-team, hits-for-the-B-team, etc.

We have attempted to codify the knowledge which is implicit in the "markers" of a game. For example, there are cyclic markers which often count "opportunities", -- strikes, balls, outs, innings. These are reset after they reach some maximum, or when applied to another event. There are also non-cyclic markers which often count "major events" such as successes and failures of subgoals and goals. The "hits-for-the-A-team" marker counts the number of successes of this sort, while the "errors-for-the-A-team" marker counts the number of failures. Clearly, the first problem is deciding the "class" of the various markers: cyclic or non-cyclic. The next problem is deciding which non-cyclic marker counts the "distinguished final goals". Identifying the general heuristic rules which would be useful in this endeavor is an open question.

In this regard, one final comment needs to be made. Namely, it may not be even possible to identify which marker counts the distinguished goals from observations alone. In baseball, both the "hits" marker and the "runs" marker count "good things"; how is a system to know the difference? A human observer, without the benefit of a tutor, would also be puzzled. The moral is "experience itself is a good teacher, but the helping hand of a tutor may still be necessary" (MIN72, WIN70).

IV.3.7 Evaluation of the Extent and Contribution of Domain Knowledge to System Performance

We must be ever vigilant lest we equip our learning system with "too much" knowledge and trivialize the learning. In light of the preceding description of the system's domain knowledge, it seems appropriate to

stop now and assess the contribution of that knowledge to the performance of the system. In this regard, three questions need to be examined:

(1) How general is the specification of the Causal-Link Schemas?

(2) How "complete" is the current set of CLSs?

(3) How tightly does this knowledge constrain the generation of hypotheses?

Answer to Question #1:

We have seen that the role of the predicates in the LHS of the CLSs is to test for properties of actions and interactions -- both at the physical interaction level and competitive game level -- rather than for the specific actions occurring in baseball (e.g., RUN, THROW). These predicates are not specific to baseball. For example, there are predicates which test for the degree-of-difficulty of an action and for a physical relationship between two actions. Thus, any action for which the system has an act-schema representation can trigger a CLS, if an occurrence of that action meets the criteria specified in a CLS. If the system were to observe sequences of actions other than those in baseball, and if the system were provided with the appropriate act-schemas, then the interpretation phase, i.e., the application of the CLSs, should proceed unhindered. In this regard, then, it does seem that the specification of the CLSs has been done in a reasonably general manner.

Answer to Question #2:

The set of CLSs currently in the system clearly have application in other games, e.g., the PHYSICAL-COMPETITION CLSs could interpret some of

the interactions in tennis, hockey, and, of course, cricket. However, whether or not the exact set of CLSs will be sufficient to interpret other games is another question. In fact, it is clear that the current set would need to be augmented in order to deal with additional aspects of baseball games, e.g., the interpretation and correlation of changes in the markers on the scoreboard with the players' goals and actions. This fact does not necessarily contradict the appropriateness of our approach to learning. What it does say is that our understanding of games is not sufficiently complete. In fact, running the system on a different game to discover where the knowledge is incomplete may be a good technique for coming to a better and broader understanding of action-oriented games.

There are ten[25] CLSs currently available to the system. With this relatively small knowledge base (c.f., [LEN76, BAR77]) the system was able to learn a number of the important concepts in baseball, e.g., the relationship between the pitcher and the opposing batter, the relationship between the various basemen and the opposing runners (see Chapter VIII for a more quantitative description of the concepts actually learned). The precise number of rules (CLSs) is unimportant; since each rule is relatively complex and since various subsets of rules have similar components, we could have broken them down into many smaller rules (see Section V.4). The important point is the identification of the "first-order" model of competitive action games into which the rules fit. While one might well take exception to the inclusion of a particular rule or to its specific formulation, and while more rules certainly need to be identified, nonetheless the rich underlying structure seems sound.

Answer to Question #3:

Independent of the number of rules in a system, an important question is the range of possible interpretations, i.e., how directly does the system proceed to the correct interpretation. While a more quantitative answer for our system is given to this question in Chapter VIII, we can comment here that several alternative interpretations (hypotheses) are put forward; in fact, some of the incorrect ones eventually become verified as truths.[26] In as much as the path to understanding which the system takes is a "branching tree", we do not feel that the small number of CLSs overly constrained the generation of hypotheses in the system's search for the correct interpretation.

One final comment is appropriate in this regard; one might take issue with the precise definition of a particular CLS or CLSs -- are the rules "tuned" to baseball? For instance, in the example in Section IV.2, one might think it ad hoc to test for DIFFICULT-ACTIONs. We take the definition of the CLSs as a serious question and have spent a great deal of time agonizing over their precise formulation. From our experiments with various earlier versions of the CLSs, we have found that changing their definitions by eliminating some tests usually results in their being applicable more often and hence generating more hypotheses. Clearly, we have tried to define the CLSs so as not to build baseball into them. The variety and number of hypotheses which are generated seems to support our endeavors.

IV.4 The Use of Acquired Knowledge During Hypothesis Generation

The true test of a learning system is its ability to use the knowledge which it has acquired. Two questions arise in this regard: how to use the information (fact or procedure) and when to use the information. By "how", we mean that a system must have a way to interpret the information. For example, if the new information is to be used as a procedure, then the system must know how to execute it; if the information is a LISP procedure, then a LISP interpreter would be appropriate. If the information is to be used as a piece of declarative knowledge, then the procedure which "reads" the fact must know how to interpret the string of symbols in the fact. By "when", we mean that a system needs to have some heuristics which suggest contexts in which the new information can be appropriately applied.

The problem of "how" to use the acquired knowledge is resolved in our system by encoding both the new information and the old information in the same representation -- as the production rules (c.f., [QUI69]). The general knowledge which is initially provided to the system hypothesizes competitive and cooperative goals and relationships; the acquired knowledge hypothesizes specific types of competitive and cooperative relationships. For example in Figures IV.1 and IV.2 we saw the general CLS PHYSICAL-COMPETITION hypothesizing a specific CLS for the pitcher-batter interaction. Thus, all knowledge packets which hypothesize goals and relationships have the same representation. From the standpoint of execution, old knowledge is indistinguishable from new knowledge; the same interpreter which applies

the general knowledge which is initially supplied can also apply the specific knowledge, which is acquired.

As an alternative to homogeneity of representation, a system could be provided with an explicit and accessible description of the representation of the acquired knowledge. This description could be used to direct the execution (interpretation) process. Such a "meta-description" of the knowledge would be comparable to the descriptions which are employed in data-base systems to define the structure and properties of the stored data (see [DAT76]). In these latter applications the use of a meta-level structural description is usually confined to the description of passive data objects; however, the existence of such an explicit description seems just as important to the definition of active procedures.

The problem of knowing when to use the acquired knowledge can be nicely resolved in a multi-leveled system such as ours. In such systems, knowledge is grouped according to function; each level contains knowledge which serves the same general purpose, e.g., in the speech understanding system HEARSAY-II [LES77] there is a level which hypothesizes words, a level which hypothesizes phonemes. In our case, knowledge about the non-intentional aspects of actions in the form of act-schemas are grouped together at one level (Level 4, Figure II.1), while the production rules which specify game knowledge are grouped together at another level (Level 5, Figure II.1). In this context the problem of when to apply acquired knowledge translates into the problem of determining into which level ought the acquired knowledge be placed. Since the hypothesized specific CLSs

serve to suggest goals and relationships for players' actions -- just like the parents which produced them -- this acquired knowledge can be placed at the same level as the parent knowledge. Thus, whenever the initial, general knowledge is applicable, the acquired, specific knowledge is also applicable.

The integration of new knowledge into the level of the parent knowledge can be viewed as a process of passing the "capabilities" [DEN66] from the parent to the spawned knowledge. Lenat [LEN76], who also adopts this technique for his system AM, reports that problems can arise if the spawned knowledge simply inherits the parent's capabilities; as the new knowledge becomes more and more specialized, the heuristics associated with general knowledge are not sufficiently constraining, so they provide little guidance in carefully choosing contexts in which the specialized knowledge would be appropriate. This problem does not arise in our system for several reasons: the situations in which the acquired knowledge is to be used themselves sufficiently constrain the range of applicability[27]; the acquired knowledge is only once removed from the general knowledge.

A typical episode in which acquired CLSs could be used in the analysis of players' actions is depicted in Figure IV.14c. There we see a member of the B-team (B1) HIT the BALL, with a player on his own team (B2) already on FIRSTBASE. Both players start RUNning. Unfortunately for the B-team, the BALL goes only as far as the opposing player (A6) at THIRDBASE, who relays the BALL to his teammate (A4) at SECONDBASE. The BALL arrives there before B2, so B2 is not allowed to execute ON SECONDBASE. A4's

subsequent THROW to FIRSTBASE arrives after B1 performs ON FIRSTBASE. If the system had previously seen, generalized, and verified CLSs for an out at secondbase (Figure IV.13b), and an infield single (Figure IV. 13a), then it could have used the acquired CLSs from those episodes instead of the general CLSs to analyze the last two competitive interactions of the episode in Figure

[(11 CATCH A4 SB BALL) - (12 WALK B2 FB)]

[(16 CATCH A3 FB BALL) - (15 ON B1 FB)]

IV.5 Choice of Representation: A Discussion

At the current stage of maturity in A.I., choosing a representational scheme is still an "art". While one seeks to match the characteristics of the domain knowledge with the expressive properties of a particular representation, there are still few guidelines to direct one in this critical process. The debate rages on as to which type of representation is best, more expressive, more natural, more efficient, etc. (see [MOO73, BOB75, BOB77b, HAY77d, WIN75] for a discussion of these issues). We shall attempt to justify our particular choices, but the reader will no doubt be able to supply reasonable counter-arguments in support of his/her favorite representation (or representations?).

The most important factor in our choice of representation was the desire to facilitate the use of acquired knowledge. As we argued in Section IV.4, we felt that encoding both the knowledge provided a priori ("old" knowledge) and the acquired ("new" knowledge) in the same represen-

tation -- whatever the representation -- would make it easier to subsequently use that new information once it was verified. This design decision affected only a subset of the knowledge in the system; since we intended that the system only learn rules which dealt with competitive and cooperative goals and relationships, the knowledge packets which dealt with other aspects of the problem (e.g., the filtering and segmenting heuristics used in Attention Focussing) did not all need to be encoded in the same representation. Thus, within each level, the representation was chosen which was appropriate to the specific task (e.g., efficient programs to perform filtering and segmenting during Attention Focussing). No difficulty was encountered in the use or integration of multiple representations (c.f., [BRO75]), since levels communicated through well-defined input/output specifications.

We chose to encode the facts (both old and new) which dealt with the hypothesis of goals and relationships as production rules [NEW73, DAV75]. We felt that this representational scheme captured the complexity of the domain knowledge quite naturally; we were able to express the conditions (left-hand side) for rule applicability as a simple conjunctive set of predicates. Moreover, each rule comprised on indivisible fact which did not need to interact directly with any other rule. Thus, the expressive facilities of the "structured objects" [NIL77] such as semantics nets [HEN76], scripts [SCHA77], frames [MIN75], data types in KRL [BOB77a] were not needed. In our system, we extended the action possible on the

right-hand side of the rule; in addition to the standard action of adding new predicates (features) to the data base (pattern descriptions in our case), the RHS was able to instantiate new production rules.

IV.6 Literature Review: Systems Which Exhibit Interpretive Behavior

The distinguishing characteristic of an interpretation task is that the input -- be it a string of words or actions, a suitably encoded visual or audio signal, or computer program -- "under-determines", in some sense, the output; domain knowledge must be brought to bear in order to "fill-in" or identify the crucial details and/or to accommodate errors in the data. For example, in our Martian example in Chapter I, we saw how they employed their model of religion in order to identify the observed activity as a type of religious ceremony. In the following, we shall not enter into a discussion of the merits of various representation schemes, but rather, we shall describe task domains in which systems exhibiting interpretive behavior have been built.

The important role of interpretation has proven to be central to the work which comes under the general heading "natural language understanding". Consider the now classic birthday party story:

> Jack was invited to Jane's birthday party. She wondered if he would like a kite. A friend told Jane that Jack already had a kite, and that he would make her take it back.

Charniak [CHA72] pointed out that in order to resolve the reference of the pronoun "it" in the above story, a large amount of world knowledge about

birthday parties, social norms, etc., needed to be employed. In addition to resolving structural problems such as pronoun reference and word sense disambiguation, world knowledge is needed in order to comprehend the meaning in the stories, where "meaning" is defined as the goals, plans and relationships of the actors in the story [SCHA77, SCHM76b]. That is, goals and plans are not always explicitly stated in a story (e.g., see Table IV.1); they must be inferred using knowledge about specific situations (e.g., the "scripts" of Schank and Abelson [SCHA77]) and general knowledge about persons' intentions, and societal constraints and norms [SCHA77, SCHM76b]. The use of such knowledge in the inference of meaning need not be limited to written text; as we argue in Chapter V, we can view the task of understanding observed actions as analogous to that of understanding written actions (stories). Thus, the techniques employed by our system closely resemble those which are used in the task of language comprehension.

Whether intentionally or not, the analogy between natural language and computer programming language is exploited by those engaged in developing systems which understand computer programs [RUT76, RIC76, SUS73, GOL74]. Namely, a model of the task domain (e.g., sorting in the case of Ruth's system and simple LOGO pictures in the case of Goldstein's) plus general knowledge about programming are used to interpret the input program in order to output a plan describing the goal of each of the statements in the code. Moreover, the task of understanding electronic circuit diagrams is also a similar problem; work has been done on providing teleological (purposive) descriptions of such diagrams [STA76, SUS77].

There are several high performance interpretive systems in the area of medical diagnosis. In the domain of blood infections and meningitis infections, MYCIN [SHO76] is given as input symptoms exhibited by a patient and available laboratory tests. Then, using its knowledge of these disease types, it presents a diagnosis and recommends a drug therapy. Alternative diagnoses (and corresponding therapies) may be suggested since the supplied data may be interpreted as supporting a number of different diseases. The INTERNIST systems [POP77] perform diagnosis in the general area of internal medicine. Signs, symptoms, and available laboratory tests are input to the system, which then uses its knowledge to analyze the data in order to output a diagnosis (or diagnoses). Kulikowski, et al., [KUL76, WEI77] have developed a system for the long term management of glaucoma; it interprets the findings of a particular patient in terms of a causal-associational network model of the disease and outputs a diagnosis, a prognosis and a therapy for the given patient.

In perception, the trend is also towards "interpretation" as opposed to simple "recognition". Given a digitized color image of an outdoor scene, the scene interpretation system VISIONS [HAN78] attempts to construct a consistent labelling (or labellings) for the important objects which it finds in the scene. While processing at the level of line and region growing is more bottom-up, stored knowledge about objects and scenes is needed top-down in order to identify regions as objects. In speech understanding, the HEARSAY-II system [LES77] is given as input an audio signal of a spoken English sentence; the goal is to identify the words in the

sentence. Processing of the signal is mediated by a host of knowledge sources, embedded in a multi-level architectural design. Here too, there is a trade-off between bottom-up processing and top-down processing; nonetheless, knowledge about the task domain (e.g., a specified domain of discourse, vocabulary, and grammar) is crucial to the success of the system.

Finally, several other learning systems employ an interpretation phase before the generalization phase. The objective of Winston's [WIN70] program is the acquisition of structural descriptions of blocks' world objects through the presentation of examples. The program is given as input a line drawing of a block's world scene; next, a set of programs transform that simple description into one in terms of objects and their relationships. The generalization phase of Winston's system -- much as in ours and in the following systems -- depends on the addition of new features to the scene description which was originally input; features such as ABOVE, SUPPORTS are needed in order to discriminate between, say, an arch and not-an-arch. The objective of Buchanan and Mitchell's Meta-DENDRAL system [BUC77] is the acquisition of rules which describe processes that transform molecular structures. In the system, the data from a mass spectrometer is input to a program called INTSUM (for INTerpretation and SUMmary) which transforms it by suggesting plausible mass spectral processes which could account for the data points in the mass spectrograph. Generalization then takes place on this latter data. The objective of Evans' system [EVA68] is the discovery of analogies between simple geometric

scenes. The input to his system is similar to that of Winston's system, i.e., simple line drawings. Features such as object type, positional relationship and similarity transformation are added to the input description. Once again, the success of the analogy discovery process, which employs generalization, depends on the previous inclusion of such features.

CHAPTER V

THE CONTROL STRUCTURE FOR HYPOTHESIS GENERATION

Chapter IV provided a static description of the knowledge which is needed for the interpretation of actions as goal-directed events in a competitive action game. In this chapter, we provide a description of the control structure which applies that knowledge.

V.1 An ATN Parser for Strings of Actions

We view the task of understanding observed actions to be similar in many respects to the task of understanding natural language. Both tasks can be represented as strings of symbols, words in the former and observations in the latter. As such, the strings have both a syntax and a semantics. There is controversy about the primacy of syntax versus semantics in natural language processing (see [WIL74]). However, we have been able to identify only the weakest of "syntactic constraints" for describing activity in competitive action games (see Section IV.3.5.1). Thus, in our task the semantics expressed in the CLSs, carries the major burden in processing.

Just as a natural language system uses a parser to control the processing of a sentence, we have designed and implemented an augmented-transition network (ATN) parser [WOO73] which serves as the control structure for hypothesis generation. However, we do not parse sentences into some syntactic representation and then apply semantic interpretation rules.

Rather, semantic rules (the CLSs) direct the parsing of an episode, which results in the immediate output of an interpretation.

An ATN grammar consists of nodes and ordered, directed arcs. In our case there are essentially two groups of nodes, those which represent states in the recognition of a competitive interaction and those which represent states in the recognition of a cooperative interaction. This is analogous to states such as noun or verb phrase in an ATN for natural language. The ATN parser operates by stepping through each action in an episode. The major arcs connecting the states in the grammar apply the competitive and cooperative CLSs to each action. The output of the ATN is an episode with pattern descriptions augmented by the firing of the CLSs and the appropriate act-schemas.

In Figure V.1 we see an infield single episode -- a "string of snapshots" -- input to the parser, while in Figure V.2 we see that episode as output by the parser. The boxes in Figure V.2 represent 'actions' while the diamonds indicate 'hypothesized goals'. The crucial shift in description from actions to goals is clearly depicted here. The relationships inferred by the act-schemas and CLSs are also represented in Figure V.2; the non-intentional interpretation (the physical enabling relationships) added by the act-schemas link action to action (box to box), while the intentional relationships added by the CLSs link goal to goal (diamond to diamond).

Consider the ATN-implemented episode grammar depicted in Figure V.3 (to make it readable, we have suppressed most of the details). The weak

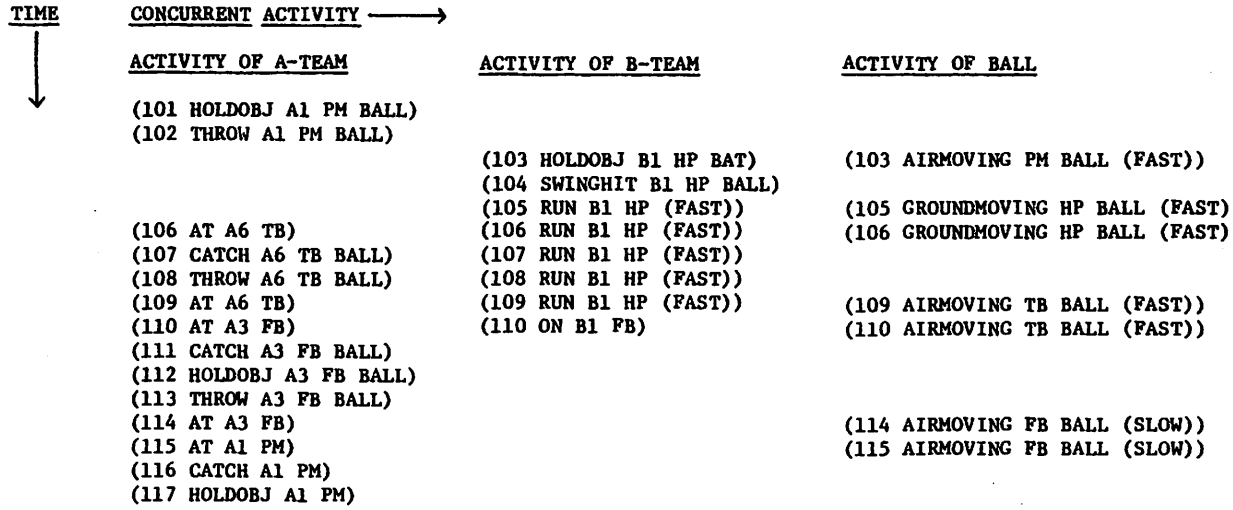


Figure V.1 An Infield Single Episode

This figure depicts an infield single. It is to be read as follows: those actions which are on the same horizontal line occur concurrently (during the same snapshot) while time marches forward in the vertical direction; those actions in the left-most column are performed by members of the A-team, those in the middle column by the B-team, and those in the right-most column by the BALL.

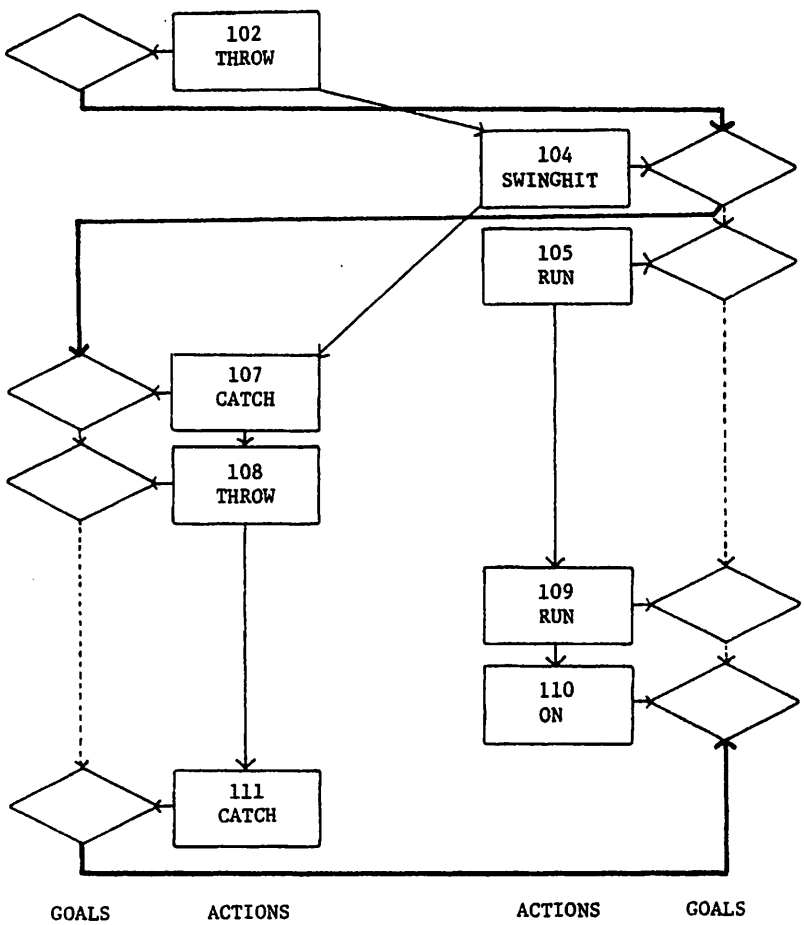


Figure V.2 Infield Single Output from the ATN Parser

The network structure depicted above highlights the various kinds of relationships inferred by the system as it processes the otherwise "isolated" actions in an infield single episode. The boxes indicate actions, the diamonds indicate hypotheses of goals, the thin lines connecting boxes (actions) indicate physical enablement relationships, the dotted lines connecting diamonds (goals) indicate hypotheses of cooperative relationships, the solid lines connecting the diamonds (goals) indicate hypotheses of competitive relationships.

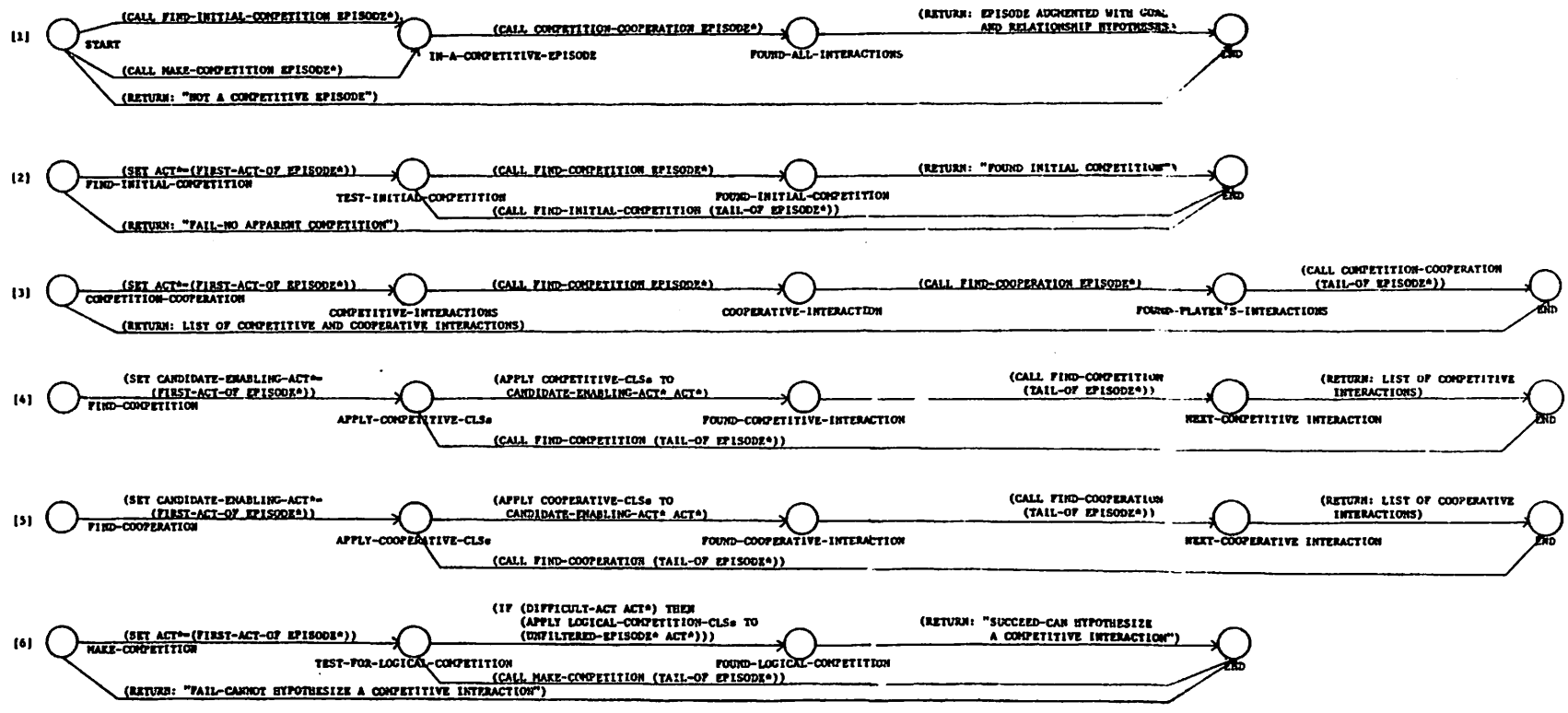


Figure V.3 The Control Structure for Hypothesis Generation: An ATN-Implemented Episode Grammar

The episode grammar, expressed as an ATN grammar and interpreted by an ATN parser, specifies the one rather trivial syntactic requirement of a competitive episode; namely, in order to be considered as a competitive episode, the episode must contain at least one competitive interaction. The ATN parser also serves as a control structure for hypothesis generation; the arcs specify when various portions of the knowledge should be applied, e.g., in subnetworks [4] the set of competitive CLSs and in [5] the set of cooperative CLSs are matched against the data.

syntactic constraint that a competitive episode must include at least one competitive interaction is expressed in the ATN grammar as the transition from the state START (network 1) to the state IN-A-COMPETITIVE-EPISODE. Two arcs permit that transition: a successful return from subnetwork 2, FIND-INITIAL-COMPETITION, which applies the competitive CLSs to the candidate actions; or, a successful return from subnetwork 6, MAKE-COMPETITION, which tries "extra" hard to find a competitive interaction (see Section V.3). For example, assume that the register EPISODE*[28] contains the infield single episode depicted in Figure V.1, and assume that the register ACT* contains the pattern description (6 SWINGHIT B1 HP BALL). A "subroutine call" will be made to subnetwork 2, since the arcs emanating from a node are ordered. The network itself (FIND-INITIAL-COMPETITION) calls subnetwork 4, FIND-COMPETITION, which starts cycling through successive actions, applying the CLSs to determine which acts (if any) could have "competitively-enabled" the above act. The conditions on the PHYSICAL-COMPETITION CLS will be satisfied by the act (2 THROW A1 PM BALL), and thus this CLS's hypotheses will be asserted and the state transition permitted. A discussion of how the system copes with not finding a competitive interaction is presented in Section V.3.

After it has been established that a competitive interaction is possibly present in an episode, the next major transition directs the system to discover all the competitive and cooperative interactions in the episode, i.e., in moving from the state IN-A-COMPETITIVE-EPISODE to the state FOUND ALL-INTERACTIONS, subnetwork 3 (COMPETITION-AND-COOPERATION-INTERACTIONS)

will be called. In this latter subnetwork, competitive and cooperative interactions are hypothesized for each player in the episode. For example, in traversing the arc from COMPETITIVE-INTERACTIONS to COOPERATIVE-INTERACTIONS, CLSs hypothesize two competitive interactions for player B1 in the snapshots of Figure V.4: (1) B1's SWINGHIT is in a competitive relation with A1's THROW, and (2) B1's SWINGHIT is in a competitive relation with A5's CATCH. Thus, multiple hypotheses are often made as to the goals and relationships of an action; the ones for B1 in Figure V.4 illustrate aspects of the proposed internal structure of a competitive episode (see Section IV.3.5.2). Finally, after establishing all interactions for a player, network 3 calls itself recursively in order to process the next act in the episode.

Figure V.5 depicts multiple, conflicting hypothesized interpretations for the relationship between the batter B1 and the fielder A8 in an outfield single. In this figure, we see that three CLSs have suggested an explanation for why B1 executed ON FIRSTBASE. The CLS for the FAIL/SUCCEED version of the STATEOF-DISTINGUISHED-OBJECT competitive relationship suggests that B1 was allowed to execute ON FIRSTBASE because the BALL, which B1 had previously HIT, had BOUNCED on the GROUND before an opponent A8 had caught it; in this case then, B1 SUCCEEDed with his goal of getting ON FIRSTBASE while A8 FAILed with his goal of preventing that action. On the other hand, the conditions for triggering two other CLSs which suggested the opposite interpretation were also satisfied by this sequence of events. The CLS for the SUCCEED/FAIL version of the STATEOF-DISTINGUISHED-OBJECT

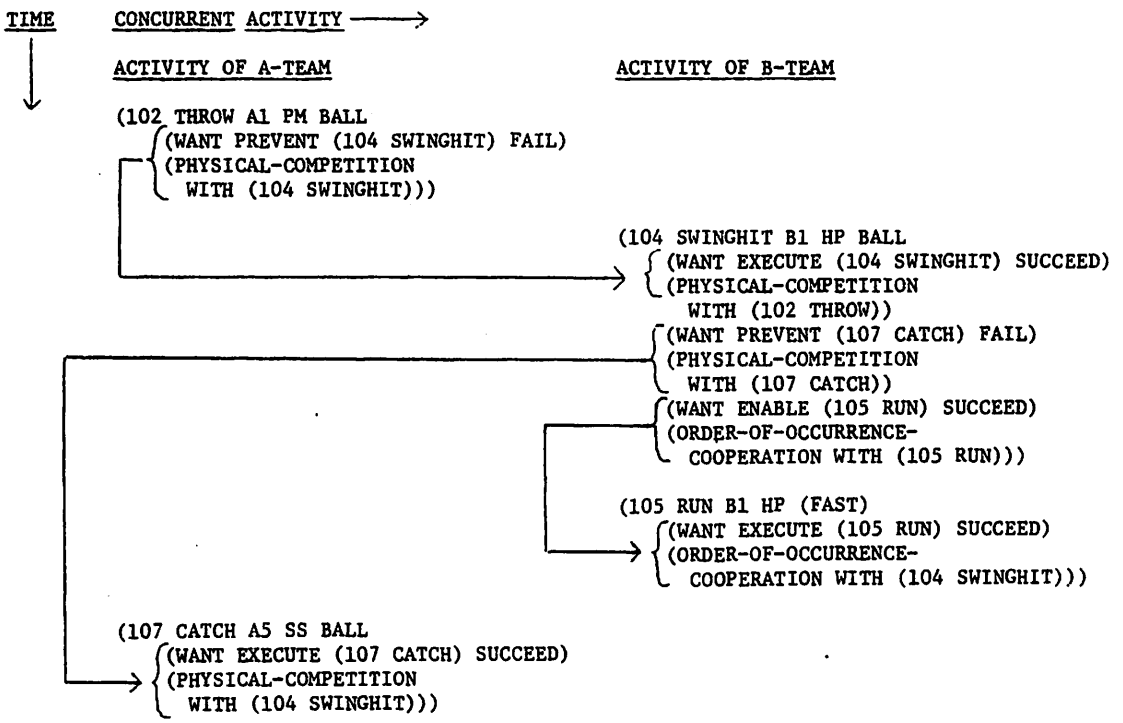


Figure V.4 An Example of Multiple Hypotheses

In this figure we see that two competitive interactions and one cooperative interaction were hypothesized as the goals of B1's SWINGHIT.

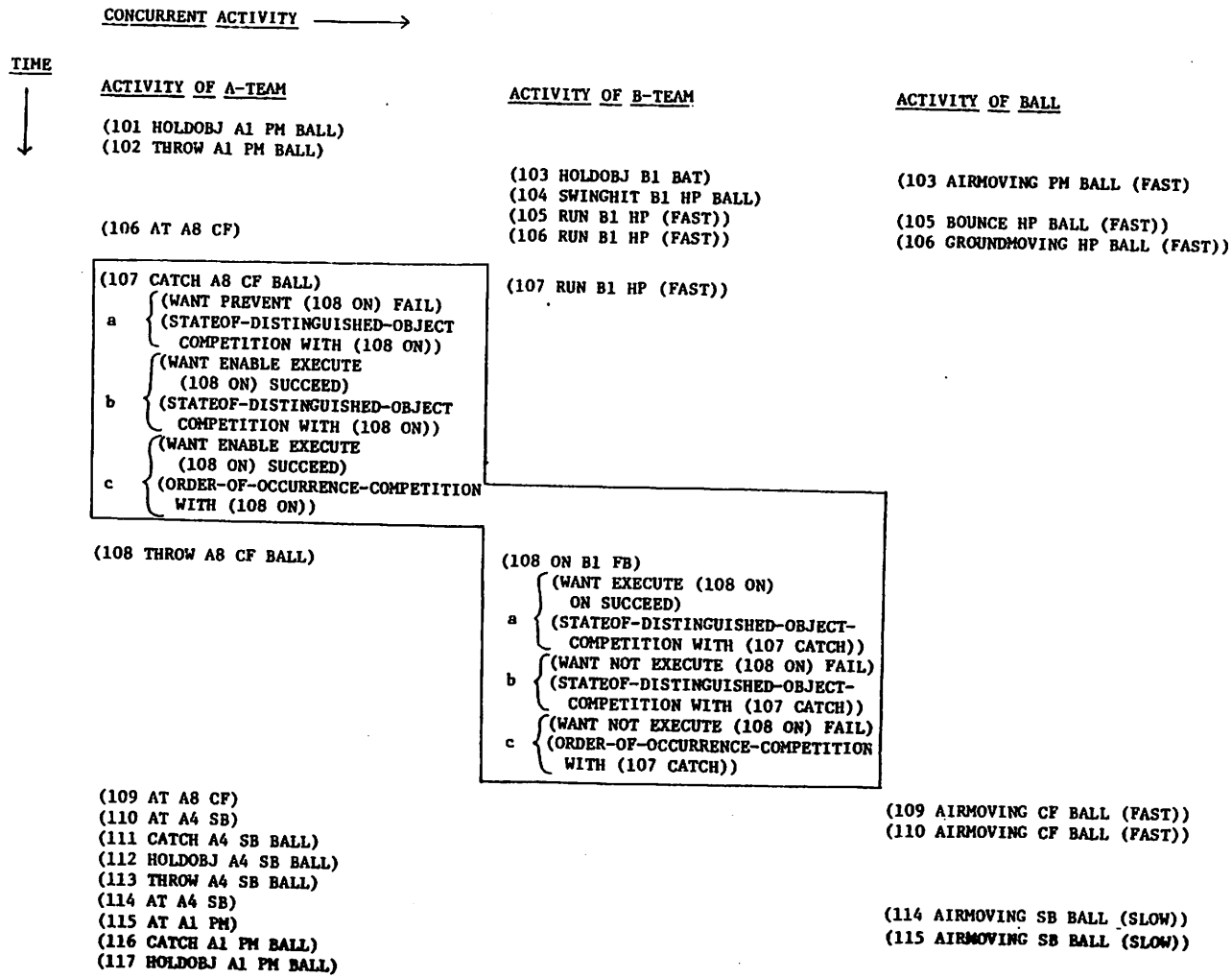


Figure V.5 An Example of the Generation of Conflicting Hypotheses

In the interaction of players A8 and B1, we see that three competitive CLSs have suggested alternative, and conflicting, explanations for why B1 changed from RUNning to executing ON FIRSTBASE.

competitive relationship and the CLS for the S/F version of the ORDER-OF-OCCURRENCE competitive relationship suggested that B1 did not want to execute ON FIRSTBASE but was forced to do so by the BALL BOUNCing on the GROUND in the case of the former CLS, and by A8 CATCHing the BALL before B1 was able to execute some other action in the case of the latter CLS. Needless to say, all these interpretations are incorrect; the first one is a necessary condition for B1 being allowed to execute ON FIRSTBASE, but not a sufficient condition. In the chapters on the Hypothesis Evaluation (VII) and Experiments with the System (VIII) we shall discuss how the system attempts to resolve conflicting -- and possibly wrong -- interpretations such as the above.

V.2 A Modified Recognize-Activate Cycle for CLS Application

As we said earlier, the CLSs are implemented as production rules. The application of the rules on the arcs of the ATN uses a modified version of the "classic" recognize-activate cycle [LEN77, DAV75]. In a "classical" production system, all the left-hand sides[29] (LHS) of the rules are matched against the data base (sometimes referred to as the short-term memory). Those rules whose conditions are true are candidates for firing. A conflict resolution strategy (e.g., see [McD77]) is used to select a candidate to fire. In our case, since more than one production rule (CLS) may be consistent with the data, we let all the rules become active whose LHSs are successfully matched. This parallel activation model of production systems is akin to the demon model of activation [SEL59].

In a classical production system, there is the notion of continued recognize-activate cycles. The idea is that a solution to a problem requires the application of several rules. However, in our case, the CLSs are self-contained knowledge packets; each CLS makes a "complete" hypothesis and does not need to refer to the effects of any other CLS. Since the CLSs are independent, and thus do not interact, only one recognize-activate cycle is needed for the analysis of each observed action.[30]

V.3 Using Knowledge to Redirect Attention

What happens if the ATN parser works its way through an episode, without finding at least one competitive interaction? Consider the snapshots in Figure V.6(a)[31] in which we see the pitcher A1, engaged in the apparently high energy action of THROWing the BALL FAST. Notice, however, that no opposing player is in the snapshots; the pitcher's teammate A2 simply CATCHes the BALL. Now consider the unfiltered (from Level 1, see Figure II.1) version of snapshots 203 and 204 in Figure V.6(b). In snapshots 203 and 204 we see the batter B1 simply standing at HOMEPLATE HOLDing a BAT. Since his actions did not change, they were eliminated by the initial crude filtering during Attention Focussing. No ordinary competitive CLS was able to operate since there was no change in one team's action correlated with a change in an opposing team's action. Thus, the ATN found itself in a state at the end of the episode in which no competitive interaction had been recognized.

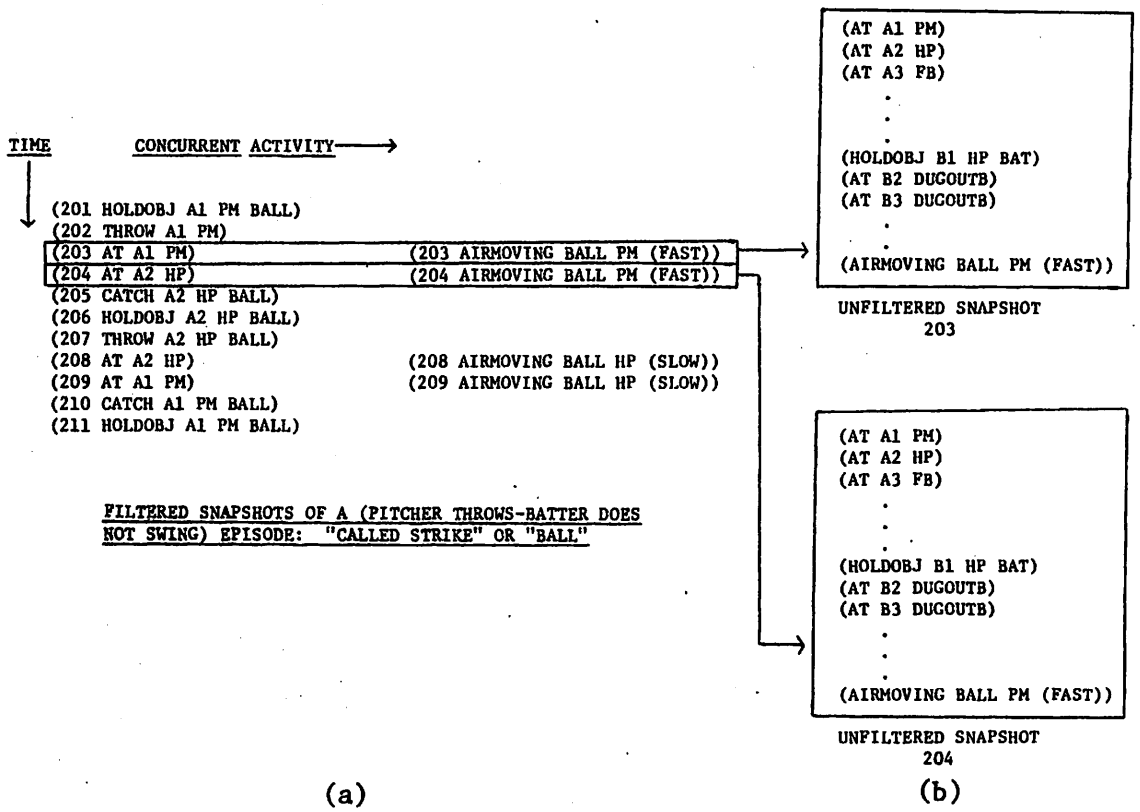


Figure V.6 Using Knowledge to Redirect Attention

Due to the initial filtering during Attention Focussing no opposing team player (the B-team) is seen in the vicinity of A1's THROW and A2's CATCH (a). In this situation no competitive CLS would fire, and thus this would not be considered as a competitive episode. However, the ATN grammar directs the system to look at the original data with the hope of making at least a weak hypothesis of a competitive relationship. The special CLSs, LOGICAL-COMPETITION, discover that B1 was standing at HOME-PLATE HOLDing a BAT, and hypothesize a possible competitive relationship between B1 and A1 and between B1 and A2.

If the system wants to believe that this sequence of snapshots is a competitive episode, then it must somehow discover a competitive interaction. To do this we equip the system with special CLSs which force the system to go back to the original data in the hope of finding an action which could be considered to be in competition with some action of the opposing team. These CLSs are only invoked when the ATN parser has passed through an episode without discovering a possible competitive interaction. Subnetwork 6, MAKE-COMPETITION, directs the system in this task. This is a case where knowledge is brought top-down to refocus the system's attention; it is as if the system were trying to find some evidence to believe what it wants to believe.

The special CLSs look in the original data for an action performed by an opposing player which is close in time and/or location to a 'difficult-action' performed by the other team. For example, in Figure V.6(b) we see that player B1 is standing in a position immediately adjacent to where the BALL THROWn by A1 is caught by A2 and in addition, B1's action occurs concurrently with team-A's actions. In this situation the special CLSs hypothesize LOGICAL-COMPETITIVE-INTERACTIONS between the pitcher A1 and the batter B1 and the catcher A2 and the batter B1. This type of competitive relationship is a catch-all relationship; the system does not know exactly what type of relationship exists, but it wants to believe that some type of relationship does exist. The version-1-CLS for this general type of competitive relationship hypothesizes a FAIL/SUCCEED outcome structure, while the version-2-CLS hypothesizes a SUCCEED/FAIL outcome

structure. For the pitcher-batter interaction, the version-1-CLS hypothesized that the batter SUCCEEDed by executing HOLD BAT at HOMEPLATE, while the pitcher FAILed by not PREVENTing the batter from performing that action; in the latter case the pitcher SUCCEEDed because he wanted the batter to simply stand at HOMEPLATE, while the batter FAILed because he did not want to execute that act. These are very weak hypotheses; there is not enough information at this point to suggest which (if either) hypothesis is true. Resolution of these conflicting interpretations again is performed during Hypothesis Evaluation (Chapter VIII) on the basis of evidence collected on similar hypotheses.

The example from Figure V.6 illustrates an additional interesting point. The actual relationship which exists between the pitcher and the batter depends on where (and if) the BALL crossed over HOMEPLATE. If the BALL crossed HOMEPLATE at a height between the knees of B1 and the middle of B1's chest, then the relationship is termed a 'called-strike', otherwise it is termed a 'ball'. Notice that this relationship depends on certain visual features which are not even observable by the system. We believe that a human observing this type of interaction would also be baffled initially; i.e., since the person was not initially looking at features such as relative-height-of-ball, the person would not be able to explain what type of relationship might be occurring. Hopefully, a person would realize that additional descriptive features are needed to explain

the observations and this would force the person to redirect his focus -- or at least ask an intelligent question. While our system makes the hypothesis that something is going on, it currently does not go back and redescribe the observations.

V.4 Strategies for Hypothesis Generation: A Discussion

This hypothesis generation strategy employed in the current version of this system is straightforward: (1) each rule (CLS) captures an entire macro concept of competitive games and thus each is independent of the others (Section V.2); (2) processing of the actions in an episode proceeds strictly "left to right" in time, with hypotheses made at one time being independent of hypotheses made at any other time. While this method of operation is successful for the problem at hand, it may prove too simplistic and too inflexible if more knowledge were added to the system. In particular, in what follows we suggest that a hypothesis generation strategy similar to that employed by the HEARSAY-II speech understanding system [LES77] and the VISIONS scene understanding system [HAN76] may be needed in an expanded version of our system.

The requirement that each CLS in the system capture an entire macro concept -- specify a competitive or cooperative relationship -- is too demanding and inflexible; some of our knowledge about competitive situations can best be described as 'fragments'. For example, consider the following rule which the current system does not employ, but which certainly is reasonable:

> When the actions of two opposing players converge at some location, this can sometimes indicate that the location is important.

Such a rule would be useful for hypothesizing that the 'location' feature is (important) relevant in an infield single where a batter is running towards firstbase while the ball is moving towards the opposing team's firstbaseman. Since this rule does not specify a complete competitive or cooperative relationship, (c.f., PHYSICAL-COMPETITION, Section IV.2) it cannot be cleanly incorporated into the system as it currently exists. Similar difficulty would arise in the attempt to include the fragmentary knowledge dealing scoreboard markers suggested in Section IV.3.5.

In order to accomodate the above type of knowledge, rules of smaller granularity than those currently in the system and a mechanism for building up hypotheses from their local inferences would need to be employed. Clearly, the problem with this approach is the difficulty of integrating local hypotheses into a consistent interpretation -- or, what is more likely, consistent interpretations. Once smaller bits of information can be utilized in hypothesis generation, the strict left to right processing should also be relaxed. This would allow, for example, hypotheses of intermediate goals to reflect information gleaned from hypotheses of final goals in an episode, as we suggested in Section IV.3.5.2. In conclusion, then, rules of smaller granularity embedded in a more flexible setting would probably be needed in order to incorporate and usefully employ the 'next increment' of knowledge about competitive action games.

Finally, two powerful notions are married through the use of an ATN parser to control hypothesis generation in our system. First, the "parsing metaphor" seemed appropriate to processing more than just strings of words, e.g., Rumelhart [RUM75] and Chafe [CHA75] have suggested grammars for stories. We have taken the next step by applying the parsing metaphor to the task of elucidating the underlying structure (relationships) in sequences of observed actions. Second, the connectedness of a network representation of the parsing mechanism was conceptually pleasing. While production rules could have been used to express the same control information as was encoded in the ATN (c.f., the meta-rules in MYCIN [DAV76]), they do not provide the same graphic description as does a network implementation of a parser. In related work, Zisman [ZIS77] has used a Petri Net to control the activation of subsets of production rules in a system which simulates the flow of information in a business office environment. The major disadvantage in the use of the ATN mechanism was the difficulty we encountered in writing and debugging grammars; if ATN's are to be more generally used, careful consideration needs to be paid to the syntax and semantics of an ATN language.[32] The notions of parsing and network connectedness were insightful in our problem, and the ATN parser was a convenient embodiment for them. However, more research needs to be done before more substantitive evaluations -- pro or con -- can be made as to the general utility of these concepts and this one method of realizing them.

CHAPTER VI

HYPOTHESIS GENERALIZATION

VI.1 Introduction

Before immersing ourselves in the details of generalization, let us first review some of the basic terminology. The process of generalization takes as input a pattern description, or a set of pattern descriptions, and strips away the irrelevant features while highlighting the relevant ones. This process has been shown to be applicable to a number of different tasks [HAY77a, EGA74, VER77b, MIN72, SOL66].

In the concept formation task, a system is presented with a set of exemplars which are instances of some general class of events (or objects). The system's objective is to abstract from the set of exemplars the common (discriminating) features for events (or objects) in the class. Figure VI.1 depicts several examples (and non-examples) of the blocks world object "arch" and the resultant general concept.

In the rule formation task, a system is again provided with a set of exemplars which are instances of some general class. However, in this task, the exemplar typically describes an event in terms of a "before state" and an "after-state". The objective is to abstract the common features in the "before-state" and in the "after-state" where the former state serves as the left-hand side of the rule, and the latter state serves as the right-hand side of the rule. This task can be viewed as essentially

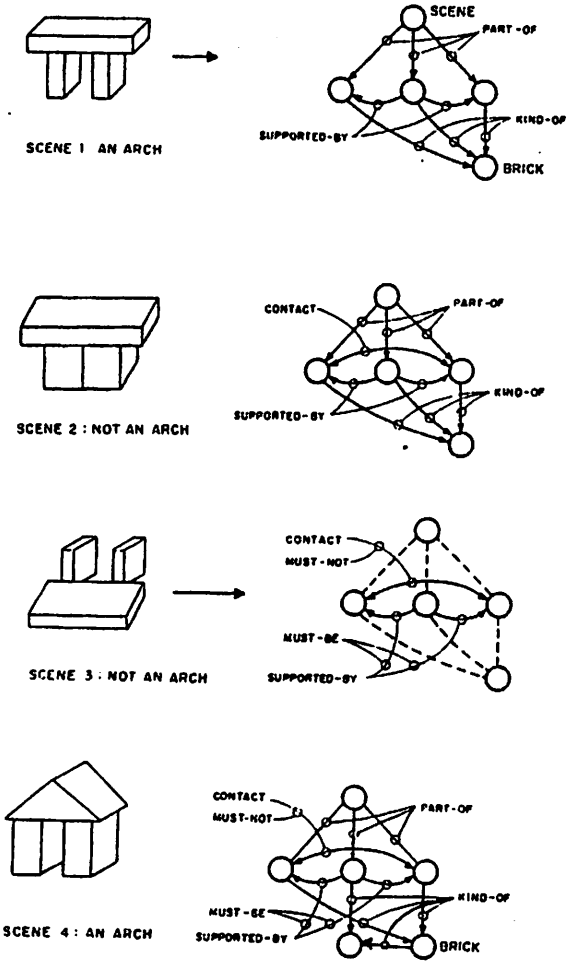


Figure VI.1 A Concept Formation Task

In the above concept formation task [WIN70], a sequence of examples and non-examples of the concept 'arch' is presented to the system. The objective is to abstract the discriminating features of an 'arch'. The resultant concept is expressed, in this case, as a network. (These are simplified versions of Winston's actual networks and are taken from [MIN72].)

a two-part concept formation task. Figure VI.2 depicts two exemplars which are the instances of the blocks world "stack" operator and the resultant abstracted rule.

In the sequence extrapolation task, a system is presented with a sequence of characters; the objective is to predict the next character (or characters) which are consistent with the underlying rule which generated the original sequence. This task can be viewed as a rule formation problem and hence can often be solved using generalization techniques.

In the analogy discovery task, a system is presented with a pair of events; the objective is to first find a general rule which maps one event into another, and then to use that rule to select a similar event from a set of additional events. This task can also be viewed as a rule formation problem. Figure VI.3 depicts an analogy problem in the domain of geometric scenes.

In this chapter we shall critically evaluate some of the previous work in generalization and describe the workings of our own system's generalization process. First we begin with a discussion of induction as it is viewed by some philosophers. Then we examine the knowledge-free, data-directed generalization (DDG) approach to the problem of discovering general rules from a set of examples. Finally we describe the knowledge-directed approach to generalization by discussing the work of several researchers and by documenting how our system employs domain

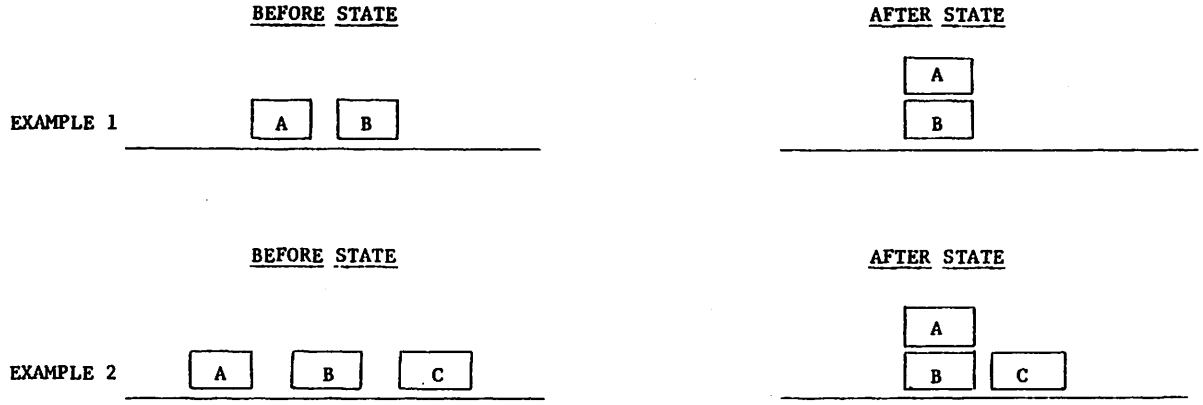


Figure VI.2 A Rule Formation Task

In the above rule formation task, the objective is to learn the rule which describes the "stack" operator. The rule is represented by two concepts, a before-state and an after-state. Concept formation is performed on the before state of all the examples, and on the after state of all the examples. The rule which would result in this case is: Before-state (blocks A and B are on the table and have clear tops) → After-state (block A is on top of block B and block A has a clear top). (Adapted from [VER77a])

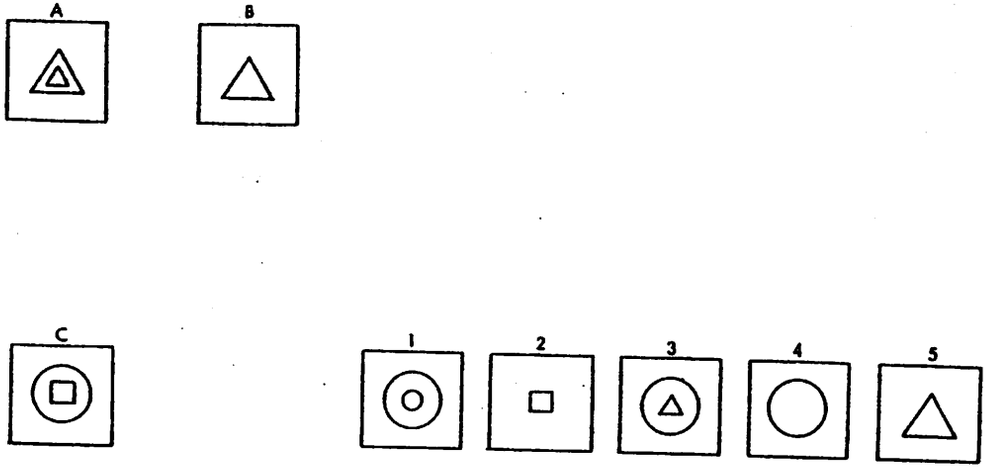


Figure VI.3 An Analogy Discovery Task

In the above analogy discovery task [EVA68], the objective is to first discover a transformation rule which maps Scene A into Scene B, and then to use that rule to map Scene C into one of the five candidate scenes. DDG is used to produce the general rules, and a domain specific evaluation function serves to order and select the best such rule; in the above example, Evans' system would choose Scene 4 as the best one to complete the analogy. (Taken from [EVA68], p. 329)

knowledge in the discovery of general rules. While systems which perform knowledge-directed generalization (KDG) often make use of data-directed generalization techniques, the key issue is that the latter techniques are subordinate to or directed by domain knowledge.

VI.2 Discovery of Rules vs. Confirmation of Rules[33]

For philosophers, the problem of induction is that of confirming, or justifying, the inference of a general rule from a set of data. The assumption is that the general rule has already been stated. Attempts are made in the study of induction to provide a set of rules which will show that an inference from specific cases to the general case is valid, i.e., that the conclusion is supported by the evidence. This is analogous to the problem of deduction. The notion of probability is relevant to the degree-of-confirmation of an inductive inference when the following rule of inference is applied:

> R_1: If PRED(X) has held in a number of instances in the past, then PRED(X) will probably hold in the future (where PRED is a predicate(s) applied to some argument X).

We leave for the philosophers (e.g., [SAL67, POP59]) a more complete discussion of the issues relating to the confirmation of inductive hypotheses: the particular relationship of probability to induction, the justification for the rule of inductive inference itself, etc. In Chapter VII, we present our own simple scheme for the confirmation of the hypotheses produced

by the learning system.

The logic of discovering the general rules in the first place is less well understood. Moreover, the possibility of such an enterprise has its detractors. We quote from Popper [POP59], who in turn quotes from Einstein [EIN18],

> However, my view of [inductive rule of discovery], for what it is worth, is that there is no thing as a logical method of having new ideas or a logical reconstruction of this process. My view may be expressed by saying that every discovery contains 'an irrational element', or 'a creative intuition', in Bergson's sense. In a similar way Einstein speaks of '...the search for those highly universal...laws from which a picture of the world can be obtained by pure deduction. There is no logical path', he says, 'leading to these...laws. They can only be reached by intuition, based upon something like an intellectual love ('Einfühlung') of the objects of experience.'

Flying in the face of such distinguished skepticism, we shall now outline two purported approaches to the mechanical discovery of general rules from specific observations.

VI.3 Literature Review: Analysis of Data-Directed Generalization as an Approach to Rule Discovery

There are a number of schemes (e.g., [HAY77c,LAR77,PLO71,VER77b]) for rule induction which, while different, nonetheless share an identifiable, common underlying theme. We shall call this approach data-directed generalization (DDG) for reasons which will soon become apparent. In this section, we shall present what we perceive is the assumption presupposed in this approach, show where this overly simply strategy goes awry, and

present mechanisms which have been employed to cope with the problems. In the process we shall be arguing that

(1) in the knowledge-free DDG approach, contrary to the claims, domain specific knowledge does enter into the mechanisms used to cope with the following factors: the combinatorial explosion of processing, the selection of a "best" set of generalizations, and the integrity and sufficiency of the sample data;

(2) domain specific knowledge must be brought to bear in order to discover rules composed of relevant features, rather than rules composed of correlated features.

The following rule underlies most of the data-directed rule discovery schemes:[34]

R_2: If a subset of features have been common to a number of instances in the past, then that subset will probably be common to instances of that class in the future and hence will aid in the recognition/discrimination of that class in the future.

Figure VI.4 illustrates how R_2 is used to actually construct generalizations. In that figure two geometric scenes are depicted; in Scene 1, there is a small, black circle and a square, while in Scene 2, there is a black square and a circle. Symbolic descriptions are also provided in Figure VI.4.[35] Generalization 'A', produced using R_2, reflects the commonalities exhibited in both scenes; the details which differed in each scene were eliminated. Thus, a variable, X, was substituted in the feature BLACK since the BLACK object was different in each scene. Also, the feature SMALL (P1) from Scene 1 was not included in generalization A since it had no counterpart in Scene 2. Note that these changes were required by the data. By "variabilizing" more constants and/or "dropping"[36]

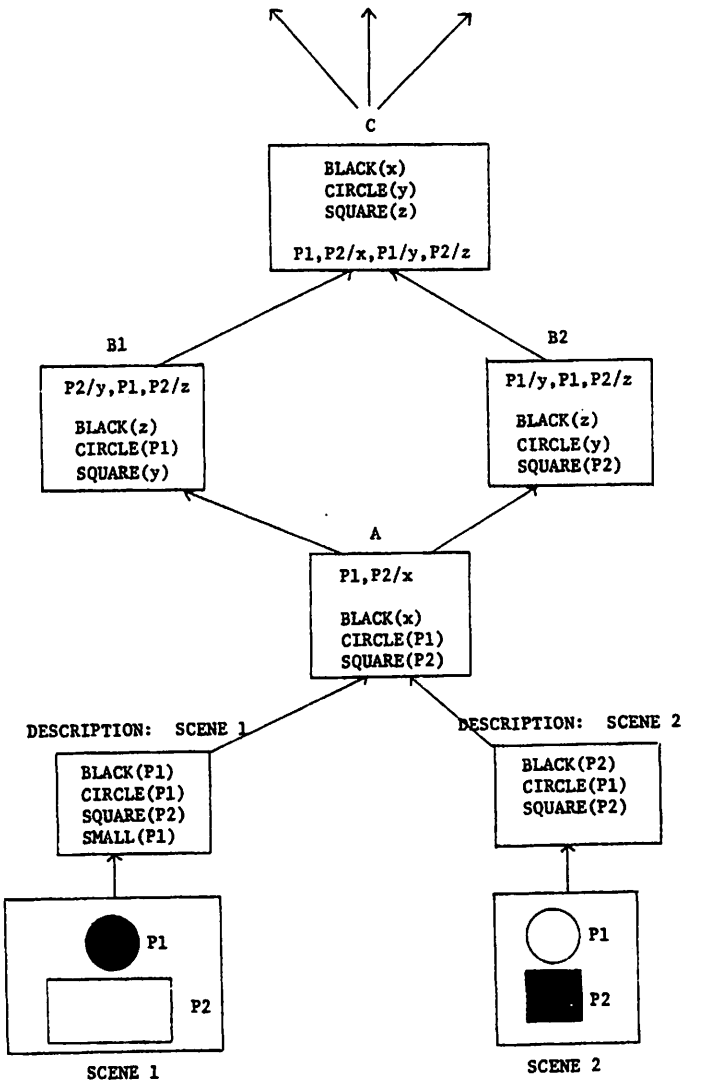


Figure VI.4 Possible Generalizations of Two Geometric Scenes

The simple-minded use of rule 'R_2' results in the production of all the generalizations for Scenes 1 and 2 depicted above. Constants are referred to by capital letters and variables by small letters; 'P1/y' means that the variable y is substituted for the constant P1.

more features, other generalizations are possible -- and are consistent with R_2. Hence the term, "data-directed generalization", appropriately describes this bottom-up technique of generalization.

Now that we have seen how R_2 can be used to create general rules from specific instances, let us explore the problems -- and their solutions -- which crop up in actual induction situations. Note that the objective of most of the work which uses a DDG rule induction scheme is the development of a system which is <u>independent</u> of the particular task domain (see, e.g., [VER75]). This claim is based on the domain independence of the inference rule, R_2, which serves as the basic mechanism for the generation of general rules from specific instances. Thus, in evaluating whether or not the suggested solutions to the problems are consistent with this claim, we must guard against the intrusion of domain semantics in the discovery process.

The first problem that must be resolved stems from the number of possible generalizations which result from the <u>simple-minded</u> use of R_2. In Figure VI.4, B1, B2, C, etc., are all legitimate generalizations of the two scenes. That is, the descriptions of both scenes are certainly instances of the general descriptions of B1, B2, C, etc. (with the appropriate substitution of variables for constants and/or elimination of features, of course). However, generalization A could be described as the "least general" one; it reflects <u>only</u> those changes required by the data and thus deviates from the data only where absolutely necessary. In the absence of some domain specific bias, a conservative rule induction

(generalization) strategy might do well to choose the "least general" generalization where possible.

The mechanism generally employed to affect this particular selection [PLO70,REY70,HAY76,VER75] is the definition of an (partial) ordering on the set of generalizations. The natural ordering is the "instance-of" relation; by appropriate substitution of variables for constants, Scene 1 and 2 (Figure VI.4) can be seen as "instances-of" the generalization A. Similarly, generalization A, Scene 1, and Scene 2 are all "instances-of" generalization B1. The key aspect to the ordering is the mechanism for producing substitutions. Reynolds [REY70] calls the algorithm which constructs the least common generalization (LCG) the "Anti-Unification Algorithm" to point out the fact that finding the LCG is the dual to constructing a most general unifier (MGU), which is used in deductive theorem proving [ROB65].

While the partial ordering suggested above can be imposed on data independently of the domain, the need has been recognized to permit a user to specify the ordering so as to take advantage of some known facts about the specific domain. For example, consider the following task: a system is supplied with a number of examples of some disease. Some of the features describing the disease are results of laboratory tests or surgical procedures which might be costly and/or dangerous. One might want to assign a value to a set of descriptors on the basis of such "costs". If so, one might then want to order the possible generalizations generated by R_2 not on the basis of the "instance-of" relation, but rather on the

basis of cost and the ≤ (less-than-or-equal) relation. Alternatively, one might want to order the generalizations on the basis of frequency of features occurring in the generalizations, etc. As far as we are aware, only Michalski's [LAR77] system allows the user to specify varying order criteria.[37]

By its very nature the "instance-of" relation only imposes a partial ordering on the generalizations; thus multiple LCG's are possible. Figure VI.5 depicts just such a situation. The features in Example 1 (Figure VI.5) can be matched consistently with two different subsets of features in Example 2; this results in two equivalent generalizations. Moreover, there may be a great number of them. A practical rule induction scheme must have some mechanisms to cope with the problem of choosing a "best" set of LCG's. Hayes-Roth's [HAY76] system and Michalski's [MIC77,LAR77] system allow the user to set parameters in their systems which reflect the user's understanding of the domain in order to control this combinatorial explosion. For example, if the set of LCG's consistent with the exemplars reaches some user specified limit, heuristics are used to select only a subset of them to keep for matching against subsequent exemplars.

The actual computational cost of matching exemplars together (or against a generalized rule) can be quite high. As Hayes-Roth [HAY76] points out, if there are n features in one exemplar and m features in another exemplar where $n > m$, then the search for possible substitutions could requires as many as $\binom{n}{m}.m!$ comparisons. To combat this combinatorial explosion, the systems of Hayes-Roth and Michalski permit the user to specify heuristics which guide the system in this search process.

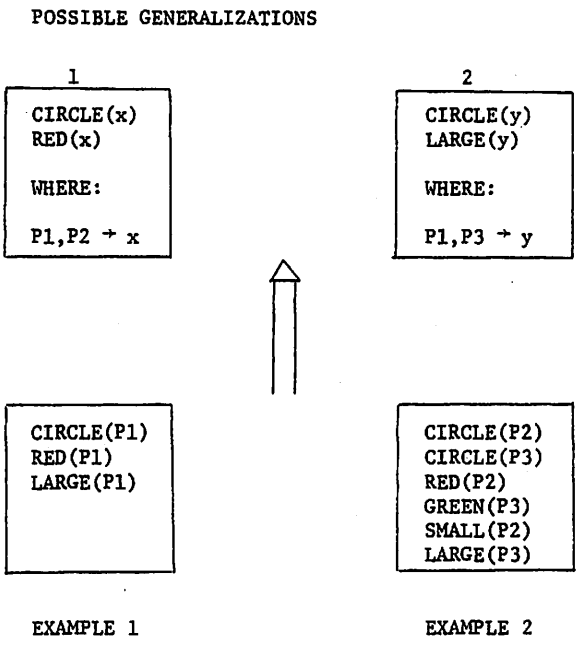


Figure VI.5 Non-unique Least Common Generalizations

There are two possible substitutions: P1 and P2 for x, or P1 and P3 for y. This results in two equivalent generalizations.

There are two types of situations in which a learning system typically operates. In "learning with a teacher", an explicit teacher is employed to oversee the learning system's progress; the teacher presents the learning system with a set of examples which have already been partitioned into general classes. Negative examples -- instances which are not in the class -- may also be presented. The system must then discover the relevant or discriminating features for each class. (Equivalently, the teacher may provide the system with feedback as to the correctness of the system's classifications.) In "learning without a teacher", no explicit teacher is employed. In this case, the system must partition the data into classes itself and discover the allowed variability within a class.

The systems by Hayes-Roth [HAY76], Michalski [LAR77], and Mitchell [MIT77] operate with a teacher. Additionally, they require that each exemplar manifest the underlying rule; if an exemplar is missing a relevant feature, then in the process of making the general rule consistent with the exemplar that feature will need to be eliminated from the general rule and hence will be lost. Effectively, this constraint requires that the trainer identify the important examples which contain the relevant (discriminating) features. DDG, then, is a "subtractive process", stripping away irrelevant features. "Noisy" data, such as that encountered by a child in the actual acquisition of language, is thus not allowed. Without the benefit of domain knowledge, these seem like reasonable and necessary assumptions.[38] While Mitchell's [MIT77] and Michalski's systems will generate rules independent of the order in which they occur, a training

set which is composed of increasingly more complex exemplars is often required.

In general then, it is clear that systems which employ the DDG approach depend heavily on a teacher who structures the problem by partitioning the data into classes and who provides a rich and graded set of training examples. These tasks require the use of domain knowledge. Thus in large measure the teacher is responsible for the success of a system which relies solely on a domain independent generalization technique such as DDG.

The above problems are exacerbated in the situation where there is no explicit teacher and the system itself must partition the data in classes. Figure VI.6 illustrates only a few of the different possible partitionings of the same data into classes. All the partitionings and classes thus produced are consistent with R_2. Therefore, again some additional information needs to be introduced in order to choose a particular one -- and avoid collapsing all the examples into one class (Figure VI.6 (d)). Analogous to the case in which all the exemplars were considered to be in the same class, we can try to impose a partial ordering on the partitionings (generalizations) and then select out one of the partitionings. For example, if we ordered the partitionings by the number of descriptors in each generalization, and then selected the partition in which the number of descriptors in each generalization were the same, then the first partition in Figure VI.6(a) would be chosen. However, this particular ordering is totally ad hoc; one would not expect it to be applicable in <u>all</u> problem domains in the same way that the ordering based on the "instance-of" rela-

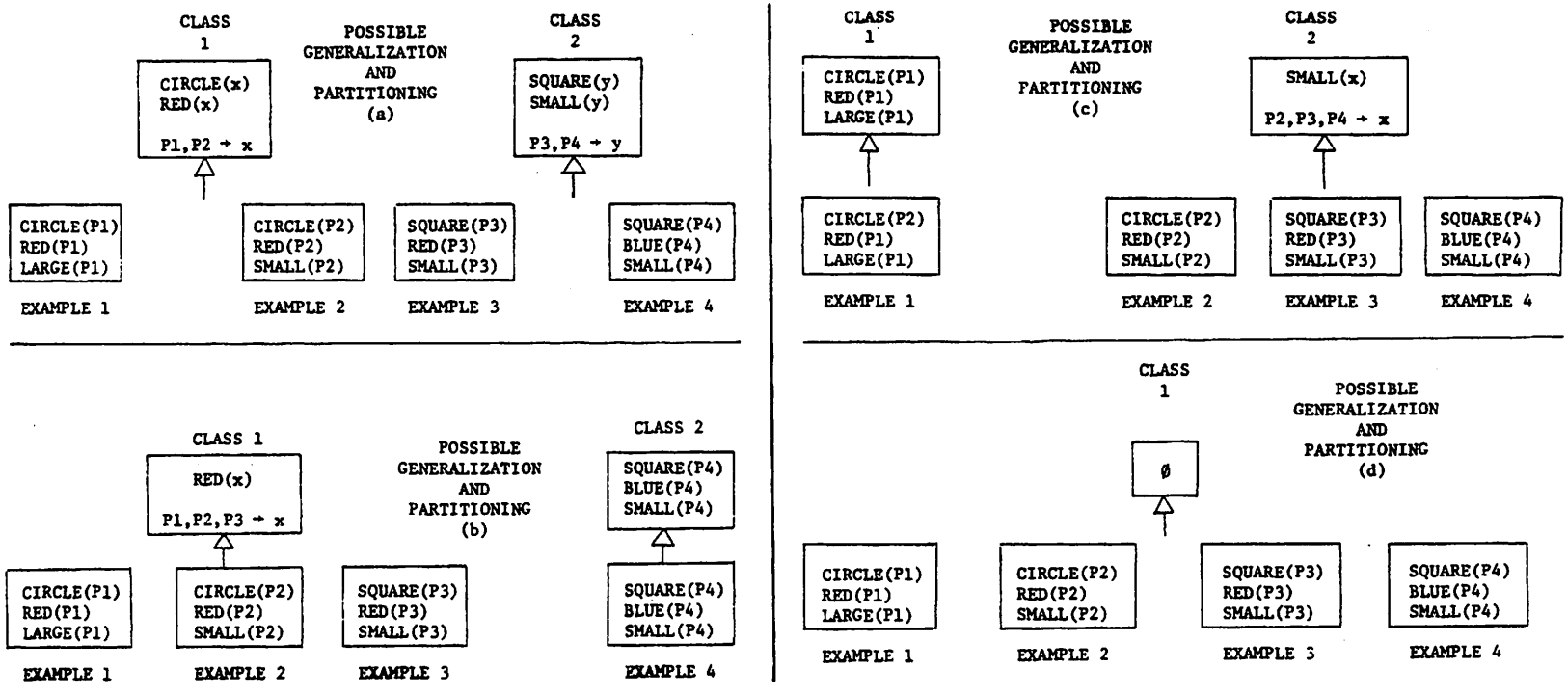


Figure VI.6 Learning Without a Teacher

In the situation where a teacher does not partition the examples into classes, the system must attempt to do so itself. In the above figure, we see four partitionings of the same four examples; more partitions are possible. Partition (d) illustrates the case where the drop rule has eliminated all the descriptors.

tion was used in all problem domains. If indeed it discovered the classes which a teacher would have provided, it would be fortuitous, i.e., such an outcome would be independent of any property of the discovery system. In otherwords, the open question is "does there exist a domain independent ordering for partitionings analogous to the role played by the domain independent 'instance-of' ordering and a selection criteria analagous to the least common generalization?" We are not confident of this question being answered in the affirmative.

In the above we have tried to identify the common problems with DDG and describe the techniques of a number of different researchers designed to cope with them. Now we shall like to comment on some other important aspects of their work. Hayes-Roth [HAY76] presents a nice characterization of a series of learning tasks (rule induction tasks) of increasing complexity. He then goes on to provide flexible and effective algorithms, and a rich set of representations (ACORNS) to cope with problems arising in those tasks. Finally, he describes several problems solved by his system, e.g., learning linguistic transformation rules.

While Plotkin [PLO70], Vere [VER77a], Reynolds [REY70] and Michalski [MIC77] all present their DDG schemes in a carefully worked out mathematical formalism, only Michalski seems to have considered in published work the computational necessities of carrying out the rule discovery process. Moreover, Michalski's formulation seems most flexible, e.g., he has constructed a whole series of programs, grounded solidly in his formalism, for use in computer-aided induction [MIC77]. Finally, Vere's [VER77a]

definition of the partial ordering on generalizations is more constrainted than the others; the program based on his characterization would not find least common generalizations[39] where other systems would.

As we have seen, then, domain knowledge does seem to be required by an approach to rule discovery which claims to be domain independent. While inference fule R_2 seems to be useful, domain knowledge was needed at each crucial point: (1) in selecting a representative set of exemplars on which to train the system; (2) in providing domain specific heuristics which were used to cope with the combinatorial explosive computational processes; and (3) in choosing a "best" generalization, where there were multiple least common generalizations.

VI.3.1 Relevance is Relative

While the above criticisms of the data-directed generalization approach to rule discovery serve to make us suspect of its utility, there is an even more fundamental problem with it. Namely, can such an approach discover rules composed of the "relevant features" -- as opposed to the merely "correlated ones"? After all this is the most important aspect of the rule discovery problem. But note that this is not the stated goal of most of the researchers which employ the DDG approach. Rather, their expressed goal is to simply find a generalization of the data. Of course, the assumption is that application of R_2 together with some domain independent constraints will in fact cause the relevant features in the data to surface. Moreover, there seems to be an implicit belief -- and desire -- that a domain independent approach should exist.

For two reasons we feel these assumptions to be dubious. First, in the early days of AI there was the same search for a domain independent mechanism for problem solving. However, experience with systems such as GPS and problem solvers based on theorem provers has shown that a great deal of task specific knowledge must be incorporated into the system in order to achieve a high level of performance (e.g., [FEI77]). Based on this experience, why should one expect there to be a problem independent method for generalization which can operate effectively without the aid of task specific knowledge?

The second reason for questioning the basis of the DDG approach stems from a closer analysis of the notion of "relevance". Namely, relevancy cannot be determined in a vacuum; it is relative to some task and problem domain. For example, those features of the pattern descriptions of the events in baseball which are relevant are those which refer to goals and relationships; the blue color of a player's eyes is not relevant to the system's objective even though this feature may continually recur. Minsky's [MIN63] quote in Chapter I makes this precise point. Thus, we make the following observation:

> Relevance is relative to the task domain and to the objective of the system.

A corollary of the above observation is:

> Relevance is not simply correlation.

The DDG approach cannot get beyond correlation unless equipped with task specific knowledge. While we know from our daily experience that many things appear correlated, we also know that many of these relationships are actually mere coincidence.

In our learning system, the objective is to use the acquired knowledge in the action game domain of baseball to (1) recognize instances of actions and goals and to (2) hypothesize additional action-goal relationships. Therefore, those features which are relevant to this task are the ones on which the system needs to base its classification scheme. No amount of data will provide information as to which features are relevant to our task. That information must come "top-down" from stored knowledge in or prior experience of the learning system itself. This is a strong conclusion. Nonetheless it appears unavoidable, if we take seriously the hypothesis that the objective of a learning system is to discover the relevant or meaningful rules, and not just some general rule. Finally, a quote from Einstein [EIN36] seems appropriate,

> There is no inductive method which could lead to the fundamental concepts of physics.

As a final observation on the distinction between hypothesis discovery and hypothesis confirmation, notice that rule R_2 is very similar to rule R_1. Roughly speaking, R_2 states that those features which are common to a number of exemplars in the past will be useful in recognizing other exemplars in the same class in the future, while R_1 states that the probability of some features appearing in an exemplar in the future is a function of the number of times they appeared in the past. It seems that R_2, which is being used in the discovery of inductive hypotheses, is just a modification of R_1, which was originally used in the justification of inductive inferences. Philosophers have worried about the justification

of R_1 -- why should R_1 be useful in establishing the credibility of an inductive hypothesis and how can R_1's validity be established. However, to our knowledge, no similar analysis has been put forth for R_2. We would dare say that the above described standard distinction which philosophers have made about the two aspects of induction has been blurred in the case of the aforementioned rule discovery systems, and that R_1's credibility was unwittingly -- and possibly unjustifiably -- transferred to R_2. Some clear thinking needs to emerge in this regard.

VI.3.2 Natural Learning Situations vs. Artificial Learning Situations

While our criticisms of the DDG approach to induction seem severe, it can be useful in some situations which we shall call "artificial learning situations". We can characterize such a situation as one in which an expert presents a learning system with scores of examples, where each example may contain great numbers of features -- all the ones which the domain expert thinks may possibly be relevant. The goal is to discover some generalizations. The data can be initially partitioned into classes, or the system may be required to do the partitioning also. This type of learning problem is purely a combinatorial problem. Since we know that the matching required to perform this task is computationally costly, the hope is that fast algorithms (possibly employing heuristics) can be developed to crunch through the data. Some researchers claim to have such rule induction systems [HAY76,MIC77].

However, the domain expert who has acquired a number of rules or abstractions about his domain has not performed that learning task via

massive combinatorial analysis. Rather, the human expert probably has acquired his understanding of the problem domain through a process not unlike the one we are developing; using semantic knowledge to greatly constrain the combinatorial search while incrementally acquiring new knowledge. This more "natural learning situation" is meant to contrast with the above "artificial learning situation".

VI.4 Knowledge-Directed Generalization (KDG): An Overview

From the discussion of data-directed generalization in the last section, we saw that there were basically two deficiencies with that approach:

(1) only rules composed of correlated features, as opposed to relevant features, could be discovered;

(2) the algorithms were effective in discovering such rules only at (possibly) great computational cost.

In order to cope with these problems, knowledge of the domain needs to be utilized. We shall now examine some systems which perform rule discovery in specialized domains and which acknowledge the important role which domain knowledge plays in controlling the combinatorial explosions and in highlighting the relevant features. In discussing these various systems, we shall attempt to abstract the methods and supporting rationales which underlie a particular system's use of domain knowledge. We shall follow this discussion with a description of the generalization process in our own learning system. As always, we ask that the reader carefully scrutinize the contribution of the domain knowledge in the system.

VI.4.1 Literature Review: Systems Which Exhibit KDG

At the end of Chapter IV, we mentioned that the learning systems of Winston [WIN70], Evans [EVA68], and Buchanan and Mitchell [BUC77] employ an interpretation phase before generalization takes place. These systems assume that after this phase all the features which could possibly be relevant will have been incorporated into the descriptions. The goal of generalization, then, will be to filter out those which are irrelevant and to highlight those which are relevant; this process of filtering and highlighting is under the direction of domain knowledge.

Recall from Section IV.6 that Winston's system attempted to build generalized structural descriptions of blocks world constructions from exemplars. While his system employed the data-directed generalization approach, it did not usually generate multiple least common generalizations. While the actual details are more complicated, it seems that one of the key factors in this regard was the use of an importance ordering[40] of the features; this ordering which was initially provided, directed the matching of one exemplar against a template (or another exemplar). To see how this scheme works, consider again exemplars depicted in Figure VI.5 and assume that the system is told that the feature "object-shape" is most important, the feature "object-color" is the next most important, and the feature "object-size" is least important. Using this ordering, a system would generate generalization 1 and not generalization 2 (Figure VI.5). In general, domain semantics would be needed in order to determine an appropriate ordering. The other key factor in the system's success was

the requirement that the trainer present a graded set of exemplars replete with "near misses" -- exemplars which do not illustrate the concept being learned due to some important difference. Taken together, these two factors limited the combinatorial search (subgraph matching) and served to highlight the most important features.

A typical problem presented to Evans' [EVA68] analogy discovery system is depicted in Figure VI.3; the task is to discover a rule which maps Scene C into one of the five candidate scenes on the basis of a rule which maps Scene A into Scene B. First, the system hypothesizes rules which map Scene A into Scene B (rule 1), Scene C into candidate-scene 1 (rule 2), Scene C into candidate-scene 2 (rule 3), etc. Then, roughly speaking, it uses DDG to create general rules from the pairwise matching of rule 1 with rule 2, rule 1 with rule 3, etc. The set of general rules produced in this way are then evaluated by a weighting function and ordered on this value. The rule with the highest value is chosen. While Evans contends that this evaluation function is domain independent, and hence imposes a domain independent ordering analogous to the "instance-of" relation, it in fact does not seem so; the evaluation function takes into account the semantics of the various features and values some features more than others. While Evans' system did employ DDG, we consider it an example of a knowledge-directed generalization system on the basis of the aforementioned domain specific ordering of rules.

The objective of the Meta-DENDRAL system [BUC77] is to discover rules which relate molecular graph structures to the processes that act on those

structures to produce characteristic mass spectrums. In Section IV.6 we discussed the important role which the system's model of mass spectroscopy played in the interpretation phase of this system. However, the rule discovery phase -- the generalization phase -- seems to employ Mitchell's DDG algorithm [MIT77]. While Mitchell does acknowledge using some domain knowledge to eliminate rules which are syntactically different but which are semantically identical, he does not emphasize the importance of this contribution. Thus it is unclear whether Meta-DENDRAL should be considered to be a KDG system or a DDG system. Buchanan and Mitchell report that Meta-DENDRAL has discovered rules in mass spectroscopy which are of publishable quality.

Waterman [WAT70] developed an interesting system which learned heuristics for betting in poker. The system's performance was at the level of an experienced human player. Waterman tested the system out in several different types of learning situations; we will examine two in particular. In "explicit learning" a trainer told the system when it had made an incorrect betting decision and supplied it with all the important information, i.e., which features were relevant, and why, in that situation. The system then had to incorporate the new information, expressed as a production rule, into the set of existing ordered production rules; the problem was deciding between modifying existing rules, or simply adding the new rule. The generalization operations used in this process were ordered in the following way; the system would first try to generalize (enlarge) the domain of the variables (features) in the rules, and then

it would try to eliminate features from the descriptions. If the number of such modifications exceeded a limit set by the trainer, only then would the system perform the third operation of inserting the new rule into the appropriate place in the ordered list. The limit was necessary to prevent overgeneralization and instability in the rules; unfortunately, Waterman only hints at the rationale that went into determining the particular setting of the limit.

In "implicit training" the system itself deduced the training information from a general model of poker and information about betting decisions which was provided. Here too, there was the problem of incorporating the correct rule into the existing set; since a rule could always be made to fire in a given situation by making it general enough, some limit on the number of features which could be generalized was provided to the system. It does not seem that Waterman's system would generate multiple least generalizations, since all that is required is the production of <u>one</u> that will work, i.e., one that satisfies the performance criteria in the training information. Thus, the factors which contributed to the high performance of the system in both situations were: the ordering of the generalization operations, the limits on the number of changes which could be made, and the training information.

Effectively, the aforementioned systems all adopted the conservative -- and data-directed -- approach to generalization: generalize only enough to accomodate the particular exemplar. While each system had its different idiosyncrasies, it would seem that almost any one of the strict DDG systems

mentioned in Section VI.3 could be modified so as to incorporate the same domain knowledge provided to each of these later systems. However -- and this is the crucial point -- the use of such domain knowledge would greatly diminish the utility of their effective, exhaustive, and combinatorially explosive search algorithms.

In contrast to the conservative approach to generalization, Fikes, et al., [FIK72] and Sussman [SUS73] adopt a more "ruthless" approach. Namely, variablize all the features in an exemplar and then use domain knowledge to re-introduce some constraints on the features. The STRIPS [FIK72] problem solving system first generates a sequence of actions which permit a robot to accomplish a task. This "plan" is then generalized by substituting distinct variables for all the constants in the plan. The same proof as was used to create the original specific plan is then carried out using this "generalized plan". This process serves to constrain the generalized plan in the appropriate ways. The proof, then, serves as the domain knowledge and supplies the constraints for this method of generalization. However, the proof procedure often does not require that a constant be substituted for a variable, or does not require that some variables be the same. Thus, the resultant generality in the plan is due to the generality left in by the proof procedure. Fikes, et al., acknowledge some problems with this technique and describe ad hoc solutions to them; however, exploiting the generality left in a proof seems like an interesting area to explore.

Sussman's system HACKER [SUS73] adopts a similar philosophy in generalizing programs which HACKER has synthesized; all constants are replaced by variables in the generalized program and the same program development sequence as was used to generate the original specific program is carried out on the generalized version. The program development sequence serves as the domain knowledge and supplies constraints on the variables (constants substituted for variables) where necessary. Sussman's procedure also may produce incorrect generalized programs; if so, then HACKER tries to "debut" them. Additionally, HACKER also uses a DDG scheme in deciding whether or not two programs are similar; programs must match on all function names, but variables can be substituted for arguments to those functions.

The systems of Winston, Evans, Buchanan, and the DDG schemes mentioned in Section VII.2 all tried to "account for the observed data". Thus the conservative approach exhibited by these systems seems reasonable. However, the systems of Fikes, et al., and Sussman were not trying to account for any data, but rather were trying to develop usable generalizations; in this context their more ruthless approach seems appropriate. We shall now examine another system which is also not constrained by any observed data.

Lenat [LEN76, LEN77] has developed a program, AM, which performs "open-ended discovery" in the domain of elementary mathematics. Discovery is viewed as heuristic search; the system is given a number of concepts (115) from finite set theory plus a number of heuristics (250) which serve as "plausible move" generators. As the discovery (search)

process proceeds, the heuristics interact with the concepts to create new concepts, fill in previously incomplete old concepts, and suggest new "moves" to be explored. The system discovered concepts in set-theory, number theory, and, with the addition of some basic concepts in geometry, also discovered additional concepts in plane geometry. New concepts in AM inherited the heuristics from the more general concepts from which they were spawned. Lenat points out, however, that it was AM's inability to also acquire new heuristics which instructed the system on when to use the newly discovered concepts that was AM's major limitation.

In order to better relate "discovery learning" to the type of learning performed by our system, we shall describe a hypothetical discovery problem using our system's knowledge which is analogous to that of AM's. Namely, the CLSs, which describe goals often found in competitive action games, and the act-schemas, which describe typical actions, would correspond in AM to concepts in finite set theory. AM's plausible move generators (heuristic rules) would correspond to a corpus of rules which described general properties of games. The discovery process would proceed by applying those rules and thus composing interesting sequences of actions and goals. In other words, instead of recognizing specific games, which our system does do, the suggested discovery system would generate specific games. Such a system might discover cricket, baseball, and various sandlot versions of baseball. The drawback to this proposed system is the current lack of a set of rules describing general properties of games. (We mentioned this problem in Section IV.3.)

VI.4.2 Knowledge-Directed Generalization in Our System

Since the goal of our system is to discover general classes composed of relevant features, domain knowledge is used to direct the discovery process; data-directed generalization plays a secondary role. Also, the sheer number of features and resulting possible matches requires the intervention of domain knowledge. Since our system does not employ an explicit teacher and training sequence, it must employ domain knowledge to help in partitioning the observed episodes into classes. We intend that this knowledge would correspond, at least roughly, to the knowledge which a trainer would need to use, if one were employed in order to segment the data into classes for a training sequence. Again, we have tried to take care not to include so much knowledge in this process as to trivialize the whole enterprise. In what follows, we shall first discuss the notion of levels of generality and its realization in our system. Then, we shall describe how the system partitions the data at various levels of abstraction into meaningful classes and discovers the allowable variation within a class. In these sections, we shall only discuss the techniques and rationales employed to perform generalization, while Chapter VIII documents the actual performance of the system.

VI.4.2.1 Levels of Generality

We know that objects or events which appear different at one level of description may actually appear the same at a higher, more abstract level of description. For example, while an infield groundout and an

outfield flyout appear different at the description level of actions, they appear similar at the description level of final competitive goals; both episodes serve to prevent the batter from reaching firstbase. At least three levels of abstraction seem natural and meaningful in action games: the competitive interaction, the episode, and the episode's final competitive goals. Clearly, some composition of episodes or some other portions of an episode (e.g., the beginning) may be meaningful in a game. However, as we indicated in Chapter IV, we do not understand games well enough in general to specify such higher level units. While we have not told the system which classes should exist at each level, we have used our understanding of the domain in order to determine that the units of activity described above do represent meaningful levels of classes. In contrast, the bottom-up DDG techniques of the previous section could not have discovered that "episodes" are a meaningful classification unit; a decision of this sort is relative to an understanding of the semantics of action games.

VI.4.2.2 Forming Classes of Competitive Interactions

Before delving into the details of our generalization scheme, and at the risk of being repetitious, we would like to review a few concepts from Chapter IV. Recall that episodes are composed of hypothesized competitive and cooperative interactions. Recall also that these interactions are represented in two forms: in the active form as production rules, and in the passive form as a set of features comprising a pattern description. It was conceptually easier to view generalization as taking

place over simple sets of features than over production rules. However, we also wanted the system to use the acquired information and integrate it easily into its existing knowledge base; this design decision dictated that the interactions be represented as production rules. Since there is an obvious translation between the two representations, we wish to make no special claims for this setup beyond conceptual aesthetics. In fact, the system employs this translation routine to keep the production rules in synch with changes (generalizations) made to the pattern descriptions. Hereafter, we shall ignore the production rule representations, and speak in terms of generalizing the pattern description representation of the competitive and cooperative interactions into classes. However, we simply remind the reader that a production rule <u>is</u> associated with each class of interactions. (See Section IV.4.)

In order to establish a general class, a subset of features of a pattern description must be distinguished. As we said earlier, a rule such as R_2 serves to pick out such a subset on the basis of commonalities evident in two or more instances. The role that this rule serves in data-directed generalization systems is analogous to the following rule employed by our system to highlight important features:

> R_3: Hypothesize as relevant that subset of features added to the pattern descriptions during Hypothesis Generation (e.g., goal, competitive relation, difficult-act, physical-enablement).

Consider, for example, the competitive and cooperative interactions of an infield single episode as depicted in Figure VI.7. Those features

```
(101 *THROW A1 PM BALL*
     *(ENABLED (103 SWINGHIT))
     *(WANT PREVENT
       (103 SWINGHIT) FAIL))
     *(PHYSICAL-COMPETITION
        WITH (103 SWINGHIT))
      *DIFFICULT-ACT
     *(CAN-EFFECT-PERFORMANCE
       (103 SWINGHIT)))
```

```
(103 *SWINGHIT B1 HP BALL*
     *(ENABLED-BY (101 THROW))
     *(WANT EXECUTE
       (103 SWINGHIT) SUCCEED))
     *(PHYSICAL-COMPETITION
        WITH (101 THROW))
      *DIFFICULT-ACT
      *CHANGED-ACT)
```

```
(103 *SWINGHIT B1 HP BALL*
     *(ENABLED(104 RUN))
     *(WANT EXECUTE
        (103 SWINGHIT) SUCCEED)
     *(ORDER-OF-OCCURRENCE-
         COOPERATION WITH
       (104 RUN)))
```

```
(104 *RUN B1 HP (FAST)
     *(ENABLED-BY
        (103 SWINGHIT))
     *(WANT EXECUTE
        (103 RUN) SUCCEED)
     *(ORDER-OF-OCCURRENCE-
         COOPERATION WITH
        (103 SWINGHIT))
```

```
(103 *SWINGHIT B1 HP BALL*
     *(ENABLED (106 CATCH))
     *(WANT PREVENT
       (106 CATCH) FAIL)
     *(PHYSICAL-COMPETITION
        WITH (106 CATCH))
      *DIFFICULT-ACT
     *(CAN-EFFECT-PERFORMANCE
       (106 CATCH)))
```

```
(106 *CATCH A4 SB BALL*
     *(ENABLED-BY (103 SWINGHIT))
     *(WANT EXECUTE
       (106 CATCH) SUCCEED)
     *(PHYSICAL-COMPETITION
        WITH (103 SWINGHIT))
      *DIFFICULT-ACT
      *CHANGED-ACT)
```

```
(107 *THROW A4 SB BALL*
     *(ENABLED (109 CATCH))
     *(WANT EXECUTE
       (103 CATCH) SUCCEED)
     *(PHYSICAL-COOPERATION WITH
       (110 CATCH)))
```

```
(110 *CATCH A3 FB BALL*
     *(ENABLED-BY
        (107 THROW))
     *(WANT EXECUTE
        (110 CATCH) SUCCEED)
     *(PHYSICAL-COOPERATION WITH
        (107 THROW)))
```

```
(110 *CATCH A3 FB BALL*
     *(WANT PREVENT
       (109 ON) FAIL)
     *(ORDER-OF-OCCURRENCE-
       COMPETITION WITH (109 ON))
      *DIFFICULT-ACT
     *(OCCURS-AFTER
       (109 ON)))
```

```
(109 *ON B1 FB
     *(WANT EXECUTE
       (109 ON) SUCCEED)
     *(ORDER-OF-OCCURRENCE-
       COMPETITION WITH (110 CATCH))
      *NOT-DIFFICULT-ACT
     *(OCCURS-BEFORE
       (110 CATCH)))
```

Figure VI.7 Competitive and Cooperative Interactions in an Infield Single Episode

The features which are asterisked in the above interactions are hypothesized to be relevant during Hypothesis Generalization; they are just the ones which the CLSs referenced during Hypothesis Generation.

which are asterisked in the pattern descriptions are those which satisfy R_3; namely, the features which are added by the appropriate act-schemas and by the CLSs which have hypothesized competitive relationships between the actions of opposing players or cooperative relationships between members of the same team. Eventually, the system will form a class for each competitive interaction depicted in Figure VI.7 on the basis of features selected by R_3.

R_3 directs the system to employ knowledge in a "top-down" fashion in order to hypothesize feature relevancy. In contrast to the DDG schemes which require that a number of examples be observed before feature correlation can be established, R_3 needs to see only one example of a competitive (or cooperative) interaction in order to hypothesize feature relevance. In effect, the most crucial aspects of a class description are determined during the interpretation phase (Hypothesis Generation).

While features are hypothesized to be relevant after observation of only one example, a class is not formed until the system sees another similar interaction. Two interactions are <u>similar</u> if they agree (match) on all the features hypothesized as relevant by R_3. Consider the final competitive interactions taken from two infield groundouts (Figure VI.8a and b) and two infield singles (Figure VI.8c and d). Two classes of final competitive interactions can be generated since the competitive interaction from the infield groundout matches a competitive interaction from the other infield groundout, but does not match either of the competitive interactions from the infield singles, and vice versa. In other words, class formation

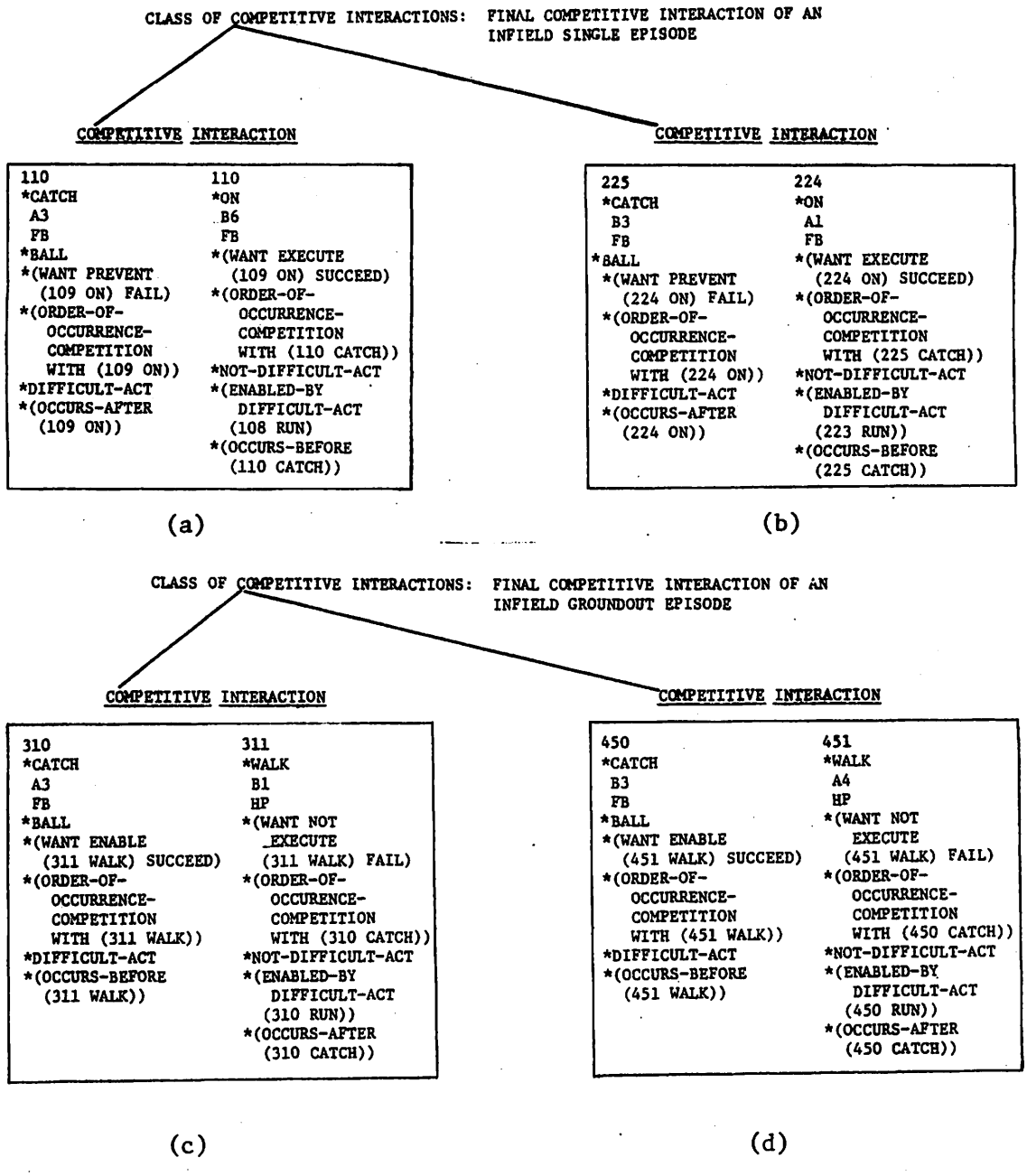


Figure VI.8 Classes of Competitive Interactions

The four competitive interactions depicted above can be partitioned into two classes on the basis of those features hypothesized as relevant (asterisked); (a) and (b) are grouped together since they both have in common the asterisked features, and similarly for (c) and (d).

in our system is a two step process: first, a subset of features is hypothesized as relevant using R_3, and then classes are formed using a rule akin to R_2, namely,

> R_2': Merge together two pattern descriptions which have a distinguished subset of features in common.

In the previous section, we saw that R_2 was used for two purposes: hypothesis of relevant features and class formation.

The variability within a class of interactions reflects the variability of the values for those features which were not initially hypothesized as relevant, e.g., the 'location' feature and the 'player' feature. The data-directed generalization technique of variabilization is used to replace constants by variables in the generalized pattern descriptions. For example, in Figure VI.9, we see the middle competitive interactions of two infield singles; in competitive interaction (a), we see player A2 HITting the BALL to player B4 at SECONDBASE, while in competitive interaction (b), we see player B3 HITting the BALL to player A6 at THIRDBASE. In order to accomodate the differences in the location to which the BALL was HIT, the 'location' feature is variabilized; the data forces the constants SECONDBASE and THIRDBASE to be replaced by a variable. However, we do not allow that variable to match any location. Rather, the system builds a set which contains the observed constants, and constrains the variable to match only elements of that set. As more examples of similar interactions are observed, the set can be extended. The conditional expression[41] for the 'location' feature in the generalized pattern description in Figure VI.9c reflects this strategy.

CLASS OF COMPETITIVE INTERACTIONS: GENERALIZED PATTERN DESCRIPTION

```
?TIME1                          ?TIME2
 SWINGHIT                        CATCH
?PLAYER1                        ?(PLAYER2 (OPPOSING-TEAMS
 HP                                        $PLAYER1
 BALL                                      $PLAYER2)
(WANT PREVENT                   $(LOCATION (MEMBER $LOCATION
 ($TIME2 CATCH) FAIL)            '(SECONDBASE THIRDBASE)))
(PHYSICAL-COMPETITION            BALL
  WITH                          (WANT EXECUTE ($TIME2 CATCH)
 ($TIME2 CATCH))                  SUCCEED
(ENABLED                        (PHYSICAL-COMPETITION WITH
 ($TIME2 CATCH))                 ($TIME1 SWINGHIT
                                (ENABLED-BY ($TIME1 SWINGHIT))
```

(c)

COMPETITIVE INTERACTION

```
464                             465
SWINGHIT                        CATCH
A2                              B4
HP                              SECONDBASE
BALL                            BALL
 (WANT PREVENT                  (WANT EXECUTE
  (464 CATCH) FAIL)              (465 CATCH)
 (PHYSICAL-COMPETITION           SUCCEED)
   WITH (465 CATCH))            (PHYSICAL-
(ENABLED                          COMPETITION
 (465 CATCH))                     WITH
                                  (464 SWINGHIT))
                                (ENABLED-BY
                                 (464 SWINGHIT))
```

(a)

COMPETITIVE INTERACTION

```
932                             935
SWINGHIT                        CATCH
B3                              A6
HP                              THIRDBASE
BALL                            BALL
 (WANT PREVENT                  (WANT EXECUTE
  (935 CATCH)                    (932 SWINGHIT))
   FAIL)                         SUCCEED)
 (PHYSICAL-                     (PHYSICAL-
   COMPETITION                    COMPETITION
   WITH                           WITH
  (935 CATCH))                   (932 SWINGHIT))
 (ENABLED                       (ENABLED-BY
  (935 CATCH))                   (932 SWINGHIT))
```

(b)

Figure VI.9 Variability with a Class

Since the values for the 'location' feature differ in the second pattern descriptions of the competitive interactions, a variable (LOCATION) is inserted in the generalized pattern description for that class of competitive interactions. However, LOCATION is constrained to match a location from the set of locations (SECONDBASE THIRDBASE) observed initially. The notation ?(LOCATION (MEMBER $LOCATION '(SECONDBASE THIRDBASE))) is interpreted as follows: the pattern matcher first assigns a location to the variable LOCATION (?LOCATION), and then it tests to see if that location ($LOCATION) is a member of the set of specified locations (SECONDBASE THIRDBASE). If true, then the match is successful; if false, then the matcher tries to find another location which satisfies the conditions.

The variabilization required by the data for the 'player' features reflects knowledge about players in games which is initially provided to the system; players involved in a competitive relationship must be on opposing teams. The constrained variable in the second generalized pattern description represents this fact.

We call the above generalization strategy constrained data-directed generalization (CDDG) and contrast it with unconstrained data-directed generalization (UDDG). The DDG systems reported on earlier in this chapter employ UDDG. In substituting a variable for differing constants, CDDG constrains that variable to match one of the values already observed, while UDDG places no restrictions on the variable. However, UDDG's more aggressive strategy has a tendency to produce overgeneralizations. Since the level of generalization of a hypothesis can play a role in the verification (or elimination) of that hypothesis, overgeneralization can have troublesome consequences. In Chapter VIII we devote an entire experiment to this issue; we report on the performance of both techniques with regards to their tendency to mistakenly accept incorrect hypotheses as truths and their tendency to form classes at the correct level of generalization.

The system is supplied with knowledge about the 'player' feature. In Figure VI.9, we see that four different players (A2 B4) and (B3 A6), were involved in the middle competitive interactions of two infield singles. Since the 'player' feature was not hypothesized as relevant during the interpretation phase, the value of this feature is allowed to vary at the direction of the data. Thus, in the general class formed for this interaction (Figure VI.9c), the constants (A2 B4) and (B3 A6) are replaced by two variables. However, the system knows that in a competitive interaction 'player variables' must each refer to the opposing team. Thus, the generalization in this example required that the variables substituted for the constants in the player features be constrained to match only those members on opposing teams.

Variables must be consistently substituted for constants. Figure VI.10 depicts competitive interactions from an "infield single" and a "double". A constrained variable ?(LOCATION (MEMBER $LOCATION (FIRSTBASE SECONDBASE))) is substituted in pattern description (e) for a pair of different locations (FIRSTBASE SECONDBASE) from pattern descriptions (a) and (c) respectively; however, the second pair of locations from the pattern descriptions (b) and (d) are the same as those from (a) and (c), respectively. Therefore, the constrained variable $LOCATION is substituted into the generalized pattern description (f). This notation represents the requirement that the value assigned to ?LOCATION in the first pattern description be substituted for the value of $LOCATION in the second pattern description of the competitive interaction.

The above example also illustrates another point: the nasty problem of overgeneralization. A "double" and a "single" are distinct events and they should not be collapsed together. Knowledge was not provided to the system to prevent this undesirable outcome, and thus an unfortunate ordering in the observation of these events led to the data-directed generalization of these two events. Since the system currently has no mechanisms for recovering from an overgeneralization, this error cannot be corrected.

In our system, we have adopted a constraint similar to that of Sussman [SUS73] and Vere [VER77a][42] which serves to prevent the generation of multiple least generalizations. All features in pattern descriptions

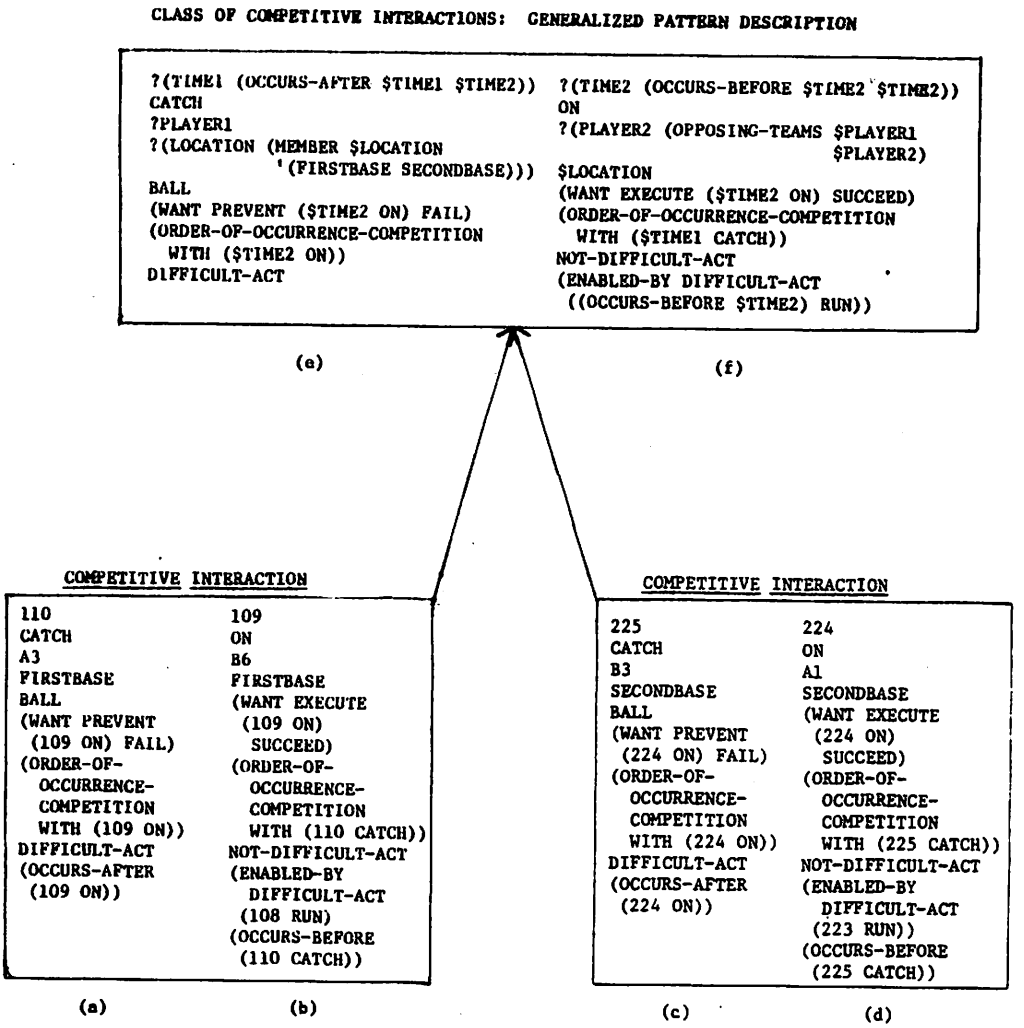


Figure VI.10 Consistent Variable Substitution

The variabilization in the generalized pattern description reflects the requirement of the data that the value for the 'location' feature be the same in both actions of the competitive interaction.

are required to participate in the generalization; no feature can be eliminated (dropped). In Figure VI.5, we saw that by matching up appropriate subsets of the features from the pattern descriptions of the two exemplars, multiple least generalizations could be generated. Thus, if only variabilization is applied to accomodate all differences in the data, only one generalization will result.

Once a generalized hypothesis is verified as being correct (see Chapter VII), then the features in the pattern description become frozen, i.e., they are no longer subject to generalization. For example, if the 'location' feature has remained constant, as it does in the

[(THROW A1 PM BALL)(SWINGHIT B1 HP BALL)

interaction, then it, in effect, has become one of the discriminating featues for interactions in that class. This is also true for features that have become variabilized, e.g., the 'location' feature in Figure VI.8. In other words, once verified, the state of a feature becomes important for the recognition of similar interactions which are subsequently observed.

Let us summarize our technique for the formation of classes by generalization. Since an explicit teacher is not employed to partition the set of interactions into classes, the key factor in differentiating one interaction from another is the use of a domain specific generalization rule such as R_3. Without it, there would be no way -- short of specifying some ad hoc generalization limit -- to prevent all the different competitive and cooperative interactions from being collapsed down into one

class whose pattern description would contain only variables in all positions. That is, the features which are added to the pattern descriptions due to the Causal-link Schemas and the act-schemas, serve as the basis for differentiating classes from each other. Clearly, the contribution of this knowledge is crucial. Nonetheless, we do not feel it compromises the performance of our learning system, since this knowledge is no more or no less than would be used by an explicit teacher in establishing the same classes. Having the system itself manipulate this knowledge requires that this information be made explicit, and thus its impact can be more easily ascertained.

VI.4.3.3 Formation of Classes of Episodes

A class of episodes is composed of a set of similar episodes, where we say that episode$_i$ is similar to episode$_j$ if and only if for all competitive and cooperative interactions there must be similar corresponding competitive and cooperative interactions in episodes i and j. Since the interactions comprising an infield single episode and an infield groundout episode are all similar except for the final competitive interaction, these two episodes cannot be merged into the same class.

The similarity requirement between episodes in a class also enables the system to bring "context" to bear on the variabilization process in order to prevent overgeneralization. That is, the variabilization of constants in the hypothesized local interactions is only to take place at the direction of other interactions which are contained in episodes of the same class. For example, the infield single and outfield single episodes

are not similar since the outfield single does not contain a cooperative interaction between a fielder and the firstbaseman which is contained in the infield single. In this case variabilization of constants is not permitted. If we were to allow generalization to take place across the interactions in these dissimilar episodes, the 'location' feature in the final competitive interactions would need to be variabilized to accomodate the differences in the two episodes. This would be most unfortunate since the fact would be lost that the fielder must be at the location to which the batter is running in order for there to be an ORDER-OF-OCCURRENCE timing relationship. While this strategy may seem ad hoc, it actually reflects a deeper understanding of the problems with generalization. Whenever global contextual constraints are available, they should be used to control generalization.

It is a strong requirement that all episodes in a class match on all the competitive and cooperative interactions. Additional knowledge about flexibility within a class might permit this constraint to be weakened, and thus a partial match [HAY77a] between episodes might suffice. For example, the system currently constructs separate classes for simple infield singles and for infield singles in which a player who is a teammate of the batter is already at FIRSTBASE. In the latter episode, at least an additional cooperative interaction is hypothesized; when the batter HITS the BALL, an ORDER-OF-OCCURRENCE-COOPERATION is hypothesized to exist between the batter at HOMEPLATE and the runner at FIRSTBASE. The requirement that all cooperative and competitive interactions parti-

cipate in the match means that infield singles without this additional cooperative interaction will be considered to be different. While this interpretation is reasonable, one might nonetheless want to lump both types together. In any case, since the system is not provided with knowledge which tells the system what can be ignored in a match, the strong requirement of complete matching is currently required.

VI.4.2.4 Formation of Classes of Final Competitive Goals of Episodes

The highest level of classification currently in the system is based on only one aspect of the episode, the goals of players in the final competitive interaction of an episode. This classification scheme allows us to merge together episode classes such as outfield flyout and infield groundout into one class, namely "outs".[43] The motivation for such a level of generalization stems from our original assumption about the hierarchical structure of plans and the special status of the final goal in that hierarchy.

In particular, the sequence of actions and subgoals leading up to the final goal can be considered as one method for achieving the final goal. However, other action-subgoals sequences may also serve to achieve the same goal. Though the episodes "outfield flyout" and "infield groundout" have somewhat different action-subgoal sequences, they can nonetheless be merged together since they both achieve the same final goal; namely, preventing the batter from reaching FIRSTBASE.

The technique for discovering classes at this level is the same as that employed at the lower level of episode classes. The "drop rule" is

used to eliminate all features except the goals hypothesized for the players engaged in the final competitive interaction. R_2' is then used to collapse together similar episodes; all episodes which match with respect to final competitive goals are merged together. Since the goal

[?PLAYER1 WANTS PREVENT $PLAYER2 ON FIRSTBASE]

[?(PLAYER2 (OPPOSING-TEAMS $PLAYER1 $PLAYER2)) WANTS EXECUTE ON FIRSTBASE]

is common to both the infield groundout and the outfield flyout, these two episodes can be merged together. A discussion of the classes formed in this manner is presented in Chapter VIII.

CHAPTER VII

HYPOTHESIS EVALUATION

Hypothesis Generation suggested an interpretation for some of the observed events in a baseball game in terms of competitive and cooperative relationships between players. Since the knowledge used in that process was general, conflicting interpretations were sometimes put forward. While inconsistencies were tolerated during that earlier phase of processing, Hypothesis Evaluation attempts to ferret out the true interpretation. In this chapter, we discuss how evidence is gathered and used to accept or reject hypotheses. First, the general problem of evaluating hypotheses in a learning environment is discussed. Next, the types of predictions used in gathering evidence are presented. This will be followed by a discussion of how hypotheses are elevated to the status of "truth". Finally, we close this chapter with a description of the complications which can arise when generalization interacts with the evaluation process.

VII.1 A Problem with Evaluation in a Learning System

In the chapter on Hypothesis Generation, we pointed out that the level of detail of the knowledge provided to a story, speech, or vision understanding system was greater than that provided to a learning system. In the evaluation of hypotheses this same observation is also true; in an understanding system, detailed knowledge about the possible hypotheses can

be built into the system. In a learning system, however, detailed knowledge might be impossible to specify a priori -- and, in any case, to do so would certainly not be legitimate. If the system is expected to make interpretations and generalizations without detailed domain knowledge, then we should not sneak detailed domain knowledge into the verification procedure.

All is not lost, however. In the next sections, we shall see that a strong model of local competitive interactions can be built from only general facts about competition in games and about the logic of causal relationships. Based on this model, predictions of events and their interpretations can be made. This allows evidence to be gathered as to the truth/falsity of the hypotheses. We feel that the knowledge included in this process does not trivialize the project, and we invite the reader's close inspection in this matter.

VII.2 The Use of Predictions in the Evaluation of Hypotheses

A strong indicator of the correctness of one's understanding is the use of that understanding to correctly predict as yet unseen events and their corresponding interpretations. In contrast, sole reliance on the recurrence of an event and its interpretation provides little evaluative information; if the first interpretation were wrong, the succeeding incorrect interpretations would only reinforce the mistake. However, by its nature, prediction goes beyond the observed event and brings additional information to bear on the question of the validity of an inter-

pretation; in our system then, the results gathered from two types of predictions serve as the basis for hypothesis evaluation.

VII.2.1 Type I Predictions: Complimentary Outcome of an Observed Competitive Interaction

Recall that in the application of the competitive Causal-Link Schemas during Hypothesis Generation, a test was made for possible "variability" in the goal outcomes of a competitive interaction (Section IV.3.2). While we hypothesized that the pitcher (A1) failed and the batter (B1) succeeded with their respective goals in the

[(THROW A1 PM BALL) - (SWINGHIT B1 HP BALL)]

competitive interaction, the system reasoned that it was hypothetically possible for the pitcher to have succeeded and the batter to have failed, with their respective goals. The possible variability of outcomes is a necessary ingredient of a competitive interaction. After all, if one player (or team) was always successful in his interactions, one would probably not call such interactions "competitive".[44] This observation on the nature of competition is the basis for the first type of prediction:

> from an observed local competitive interaction where one player succeeds and the other player fails with their respective goals, predict that the same competitive interaction can take place, but the outcomes will be reversed -- the one who succeeded will fail, while the one who failed will succeed.

In other words, an event and its interpretation can be predicted from the hypothesis of a competitive interaction; a version-1-CLS which hypothesizes a FAIL/SUCCEED outcome structure predicts an event with a SUCCEED/FAIL

outcome structure which can be recognized by a version-2-CLS for the same type of relationship, and vice versa.

Predictions, encoded as production rules, are passed to Attention Focussing where they are matched against the incoming observations. If a prediction is triggered, a message to that affect is sent back to Hypothesis Evaluation where the confidence value for that hypothesis is increased. Currently, the confidence value of a hypothesis is a simple function of two factors: the results of such predictions and the consistency or inconsistency of the hypothesis with previously acquired knowledge. However, should the system's knowledge base be expanded, the resultant increase in possible interpretations might require a more sophisticated measure of confidence for proper evaluation (cf., the certainty factors of MYCIN [SHO76]).

Examples of Type I predictions are given in Figure VII.1c. The two examples of infield single episodes give rise to the generalized predictions depicted there. A subsequent swing-and-miss episode (Figure VII.1d) triggers prediction **P1[45] based on hypotheses **H1 (Figure VII.1a) and **H4 (Figure VII.1b); the SWINGMISS (Figure VII.1d) is recognized to be an instance of (NOT SWINGHIT), the players involved are on opposing teams, and the complimentary CLS which hypothesizes the SUCCEED/FAIL outcome structure for the PHYSICAL-COMPETITION interaction was triggered. Similarly, the final competitive interaction of an infield groundout episode (Figure VII.1e) triggers prediction **P3 which is based on hypotheses **H3 and **H6. In both cases, the effect of triggering the predictions is an

increase in the confidence value for the hypotheses on which the predictions were based.

We indicated earlier (Section IV.3.5) that the goals of the participants in hypothesized cooperative interactions are trivially labelled as SUCCESSes. A more sophisticated interpretation would take into consideration the success or failure of the final competitive goals or other enabled competitive interactions. Since our understanding of how that analysis should proceed is at best sketchy, we have simply let the initial hypotheses stand. At the moment, we have no prediction rule for cooperative interactions analogous to the case of competitive interactions. Thus, currently we do not evaluate hypotheses of cooperative interactions.

The type of prediction outlined above is powerful yet independent of a specific game. It is appropriate in all games where competition is clearly defined. Thus we do not feel we have smuggled into this aspect of the verification procedure knowledge which trivializes the learning problem.

VII.2.2 Type II Predictions: What Should NOT Be Observed

The structure "if conditions X hold then Y is enabled"[46] underlies all the competitive relationships hypothesized by the CLSs. For example, assume that the system observes the following actions:

101	102
[ON B1 FIRSTBASE]	[CATCH A3 FIRSTBASE BALL]

B1 performed the act ON at time 101 while his opponent A3 caught the BALL at time 102. The CLS ORDER-OF-OCCURRENCE$_{VERSION-1}$, which hypothesizes an F/S outcome structure[47], infers that the following relationship might be true:

(1) if ([ON B1 FIRSTBASE] OCCURS-BEFORE [CATCH A3 FIRSTBASE BALL])
then [ON B1 FIRSTBASE] is allowed to continue.

The crucial feature is the introduction of the timing relation OCCURS-BEFORE[48]; the other CLSs introduce different relations in the above expression.

Since we take statements such as (1) as rules, we require that the conditions X be both necessary and sufficient for Y. Thus, if X is a necessary condition for Y, then whenever ~X is observed ~Y should be observed; if X is a sufficient condition for Y, then whenever X is observed Y should be observed. The condition of necessity implies, for example, that in the former case whenever the system observes

~([ON B1 FIRSTBASE] OCCURS-BEFORE [CATCH A3 FIRSTBASE BALL])

then it should see

~([ON B1 FIRSTBASE]),

i.e., if A3 CATCHES the BALL before B1 executes ON FIRSTBASE, then B1 should not be allowed to execute that act.

Based on these two requirements of a rule, the system makes two predictions of events that should NOT be observed: it should not be the case that if ~X is observed Y is also observed, and it should not be the case that if X is observed ~Y is observed. The former case disconfirms that X is a necessary condition for Y and the latter case disconfirms that X is

a sufficient condition for Y. For example, from hypothesis (1) the system predicts that if it first observes

(2) ~([ON B1 FIRSTBASE] OCCURS-BEFORE [CATCH A3 FIRSTBASE BALL])

i.e., A3 CATCHES the BALL before B1 executes ON FIRSTBASE, and then it observes

(3) [ON B1 FIRSTBASE]

then (1) is not a necessary condition for (3). An observation such as (3) would then have the effect of invalidating the hypothesis (1) upon which the prediction was based.

Predictions of this type are also passed to Attention Focussing where they are matched against the subsequent input. If a prediction of this latter type is found to occur, the confidence value for the hypothesis on which the prediction is based, is made to go negative. The motivation here is that 1 negative piece of evidence for a hypothesis serves to effectively eliminate it; however, positive evidence (Type I predictions) does not necessarily confirm a hypothesis, but only serves to increase the confidence in that hypothesis.

VII.3 From Hypothesis to Truth

While the satisfaction of either of the Type II predictions serves to eliminate the hypothesis on which the prediction is based, the satisfaction of a Type I prediction only serves to increase the confidence value of the hypothesis on which the prediction is based. A threshold, which is provided to the system by an external source, is then used to establish a

cutoff point for truth; as the confidence value of a hypothesis moves above threshold, it becomes accepted as a truth.

The dangers involved with the choice of a threshold are well-known [RIS76]. If the threshold is set too low, then false hypotheses may be turned into truths. If the threshold is set too high, not all the truths will be discovered. In Chapter VIII, we discuss the results obtained with our system using different threshold settings.

Once a hypothesis has been accepted as a truth, it is used to modify the confidence values of other hypotheses; those whose goals are consistent with the new truth have their confidence value increased by a constant, while those which are inconsistent have their confidence value decreased by a constant. For example, when the system accepts the hypothesis that getting ON FIRSTBASE is the intended goal of the batter in an infield single episode, then it can decrease the confidence value of an inconsistent hypothesis from an outfield single episode that says getting ON FIRSTBASE is not what was intended by the batter.

The use of acquired knowledge to aid in the evaluation of other hypotheses can be dangerous. It assumes that a hypothesis verified in one context is relevant to the verification (or elimination) of hypotheses put forward in other contexts. In order to soften the effect of this procedure, the constant by which the confidence values on the hypotheses are modified has been kept low (see Chapter VIII).

Finally, an important process in the determination of truth is the ability to recover from an error made in this regard. Currently, our

system is not equipped with such backtracking capability. If it accepts a falsehood as truth, and proceeds to eliminate other actual truths as being falsehoods, it cannot recover. An example of such a situation is presented in Chapter VIII.

VII.4 Verifying Individual CLSs vs. Verifying Episodes

While the system is trying to verify hypotheses of local competitive interactions, it is also trying to verify an "episode's worth of hypotheses". This proves difficult. It is often the case that the competitive hypotheses in an episode all have different confidence values, i.e., are at different levels of verification. Rather than wait until all of the hypotheses in an episode are verified -- a state that may never be reached -- we have adopted a weaker requirement: if the final competitive interaction is verified and if a majority of the competitive hypotheses are verified, then the episode is considered to be verified.

Once an episode has been verified, the system creates a recognition template for that episode. Such a template contains the set of competitive and cooperative interactions in an episode. Recognition templates are used by the Focussing of Attention procedure to enable it to "see" a larger unit than merely one interaction. Of course, the system must actually see the interactions that make up the template, but having a template that collects all the interactions is a first step at "hierarchical perception".

VII.5 The Interaction Between Generalization and Evaluation

The interaction between generalization and evaluation is a subtle -- and troublesome -- one. In learning systems which assume the correctness of their data this problem does not arise, since evaluation is not employed. Unfortunately, a real-world task such as the one at hand does not afford this extra simplification. That is, errors in interpretation do crop up, and need to be weeded out. The problems arise when generalization enters the picture to modify the original interpretation and the predictions based on that interpretation.

An incorrect generalization could cause a correct hypothesis to be eliminated, or an incorrect one to be verified. Overgeneralization is particularly bothersome: it increases the number of contexts in which the predictions can be matched. One implication of this observation is that incorrect hypotheses which are overgeneralized have a tendency to become verified.

Since our system currently has no error recovery procedures, our strategy has been to decrease the likelihood of an overgeneralization and thus decrease the chances of verifying an incorrect hypothesis. The conservative generalization technique described in Chapter VI is the key to this strategy. In the next chapter, we compare the performance of this technique with another, more ambitious generalization technique with regards to this issue.

C H A P T E R V I I I

SYSTEM PERFORMANCE: THREE EXPERIMENTS

VIII.1 Experiment #1: A Representative Run

In this first experiment, we present results achieved by the system which are representative of its capabilities. Since practical considerations prevent us from running the system on enough games of baseball to be statistically meaningful, we must be satisfied with this more qualitative method of evaluation. In particular, the run we have chosen to report on here exhibits the system's strengths and weaknesses. In this section we shall explore these issues in more detail by tracing the system's performance through the various levels of processing.

The data for the system -- episodes in baseball -- were generated by a computer program which simulated a simplified version of baseball composed of the nineteen episode types listed in Table VIII.1. Note that the nineteen episode types represent only twelve distinct classes. Due to slight differences in timing relationships, some classes are represented by more than one episode type. Note too, that not all episodes which can occur in baseball are represented in Table VIII.1. Two factors contributed to the omission of such events as "foul ball" and "home run": (1) our desire to simplify the problem; (2) our incomplete understanding of the underlying general knowledge needed in order to fully analyze concepts such as "home run". Nonetheless, the episodes which are observed by the

	NAME OF EPISODE	NUMBER OF TIMES IT APPEARS
1	Infield Single	3
2	Infield Groundout	8
3	Outfield Single-I	3
4	Outfield Single-II	3
5	Outfield Single-III	4
6	Infield Flyout	6
7	Outfield Flyout	9
8	Outfield Double	3
9	Out at Secondbase	3
10	Infield Single plus Baserunner-I	2
11	Infield Single plus Baserunner-II	2
12	Infield Single plus Baserunner-III	2
13	Double-play	3
14	Fielder's Choice-I	3
15	Fielder's Choice-II	2
16	Fielder's Choice-III	2
17	Fielder's Choice-IV	4
18	Swing-and-miss	16
19	Throw-and-noswing	27
		105

Table VIII.1 Episodes Presented to the System

system do constitute a significant and representative sample of events which occur in baseball.

The ordering of the episodes presented to the system is random and thus does not conform to the rules of baseball. A baseball fan would no doubt be upset by the sequence of events generated in this manner; e.g., in one episode string, a fielder's choice follows a double play. However, since our system does not possess knowledge which could link episodes together, this simplification of the game does not affect the system's sense of propriety. The episodes do contain the variability necessary to test the generalization capabilities of the system; various values for the "location", "player" and "timing" features are present in the data.

In this first experiment the threshold on the confidence value for the acceptance of hypotheses as truths is set at six. Recall (Section VII.2) that the confidence value is based on the results of predictions (Type I and Type II)[49] and on consistency or inconsistency with knowledge which has already been acquired. In particular, a correct matching of a Type I prediction contributes 2 to the confidence value of a hypothesis while the consistent matching of the hypothesis to an already verified hypothesis contributes 1.

VIII.1.1 Filtering and Segmenting the Input

In this run, a continuous "string" of 1680 snapshots was first input to a SNOBOL program which filtered and segmented the snapshots. Table VIII.2 depicts the effect on an episode and on a snapshot of

BASED ON 19 EPISODE TYPES

INPUT			AFTER FILTERING		
TOTAL # OF SNAPSHOTS	TOTAL # OF PATTERN DESCRIPTIONS	TOTAL # OF FEATURES*	TOTAL # OF SNAPSHOTS	TOTAL # OF PATTERN DESCRIPTIONS	TOTAL # OF FEATURES
311	5598	27,990	311	616	3080

*There are five features in each pattern description.

THE AVERAGE EPISODE

INPUT			AFTER FILTERING		
AVG. # OF SNAPSHOTS	AVG. # OF PATTERN DESCRIPTIONS	AVG. # OF FEATURES	AVG. # OF SNAPSHOTS	AVG. # OF PATTERN DESCRIPTIONS	AVG. # OF FEATURES
16	295	1473	16	32	162

THE AVERAGE SNAPSHOT IN AN EPISODE

INPUT		AFTER FILTERING		
	AVG. # OF PATTERN DESCRIPTIONS	AVG. # OF FEATURES	AVG. # OF PATTERN DESCRIPTIONS	AVG. # OF FEATURES
	18	90	2	10

Table VIII.2 The Effects of Filtering on the Input Snapshots

filtering out the unchanging activities at Level 2. The calculations in Table VIII.2a were based on one example from each of the nineteen episode types rather than on all 105 episodes. Recall that a pattern description for an action is composed of five features: action, actor, location, time and modifiers. The reduction in the number of pattern descriptions is most drastic; for an average episode with sixteen snapshots (time frames), the number of pattern descriptions input was 295 while the number left after filtering was only thirty-two (Table VIII.2b). The other interesting statistic is the drop in the number of pattern descriptions in a snapshot due to filtering, from eighteen to two on the average (Table VIII.2c). Note that the input to our system has already been prefiltered and minor actions such as nose scratching, foot shuffling, etc. have been eliminated. Also, in a real game there is far more movement of the fielders when a ball is hit than is present in our simulation. An interesting extension to the system would be the development of mechanisms to cope with the actual "raw data" in a baseball game.

The heuristic used by the SNOBOL program to partition the 1680 snapshots into 105 episodes stems from the observation that there are cycles of activity in games and competitive episodes are marked by periods of high activity surrounded by periods of low activity. The distribution of the various episode types in the 105 episodes is given in Table VIII.1.

VIII.1.2 The Interpretation Phase: Applying the Act-Schemas and the Causal-Link Schemas

The act-schemas and causal-link schemas currently employed in the system are listed in Table VIII.3. The ATN parser applies them to each

Act-Schemas

CATCH	RUN
HOLDOBJECT	WALK
THROW	AT
SWINGHIT	ON
SWINGMISS	

(a)

Causal-Link Schemas

Competitive CLSs:		
Version-1: Hypothesize FAIL/SUCCEED Goal Structure	Version-2: Hypothesize SUCCEED/FAIL Goal Structure	Reference
PHYSICAL-COMPETITION		Section IV.3.2.1
ORDER-OF-OCCURRENCE-COMPETITION		Section IV.3.2.2
STATEOF-DISTINGUISHED-OBJECT-COMPETITION		Section IV.3.2.3
LOGICAL-COMPETITION		Section V.3
Cooperative CLSs:		
PHYSICAL-COOPERATION		Section IV.3.3
ORDER-OF-OCCURRENCE- COOPERATION		Section IV.3.3

(b)

Table VIII.3 Summary of General Knowledge Supplied A Priori to the System

action (pattern description) in the filtered episodes. Table VIII.4a depicts the total number of activations for those "knowledge sources"; the numbers were computed on the basis of one example from each of the nineteen episode types (Table VIII.1). In accordance with the rules of baseball, we evaluated whether or not a competitive hypothesis was correct and found that 75% of them were correct. The distribution of the various types of competitive hypotheses is depicted in Table VIII.4b. The actual number of hypotheses verified or eliminated varies from run to run. In the next section we present the results taken from one representative run of the system.

On the average, 5.7 hypotheses of competitive and cooperative interactions were put forward in an episode (Table VIII.4c). In order to provide a more accurate picture of the meaningful hypotheses, the number of cooperative hypotheses ignores the rather trivial "physical cooperation" hypothesis suggested for action sequences executed by the same player. For example, the action sequence [(102 HOLDOBJECT B1 HP BAT)(103 SWINGHIT B1 HP BALL)] is not counted as a significant cooperative hypothesis.

The "bushiness" of the interpretation phase can be seen in Table VIII.4d. In that table, a "competitive interaction" simply indicates that some competitive relationship was hypothesized to exist between two opposing players; based on one example from each of the nineteen episode types there were fifty-nine such interactions hypothesized. However, sometimes more than one CLS was put forth as an explanation for a competitive interaction. That is, on the average 1.2 competitive hypotheses were suggested

(a) OVERALL STATISTICS FOR THE ACT-SCHEMAS AND CAUSAL-LINK SCHEMAS

TOTAL # OF PHYSICAL RELATIONSHIPS INFERRED BY ACT-SCHEMAS	TOTAL # OF COMPETITIVE AND COOPERATIVE HYPOTHESES	TOTAL # OF COOPERATIVE HYPOTHESES*	TOTAL # OF COMPETITIVE HYPOTHESES	TOTAL # OF CORRECT COMPETITIVE HYPOTHESES+	TOTAL # OF INCORRECT COMPETITIVE HYPOTHESES+	PERCENTAGE OF CORRECT COMPETITIVE HYPOTHESES
537	109	40	69	52	17	75%

*Ignores trivial physical cooperation hypotheses.
+Correctness and incorrectness was determined by human evaluation in accordance with the rules of baseball.

(b) DISTRIBUTION OF COMPETITIVE HYPOTHESES

TOTAL # OF PHYSICAL-COMPETITION HYPOTHESES		TOTAL # OF ORDER-OF-OCCURRENCE COMPETITION HYPOTHESES		TOTAL # OF STATEOF-DISTINGUISHED-OBJECT-COMPETITION HYPOTHESES		TOTAL # OF LOGICAL-COMPETITION HYPOTHESES	
F/S	S/F	F/S	S/F	F/S	S/F	F/S	S/F
17	18	9	13	5	3	2	2

(c) AVERAGE NUMBER OF ACT-SCHEMAS AND CLS ACTIVATIONS PER EPISODE

AVG. # OF ACTIONS++	AVG. # OF ACT-SCHEMA ACTIVATIONS	AVG. # OF COOPERATIVE HYPOTHESES*	AVG. # OF COMPETITIVE HYPOTHESES	AVG. # OF COMPETITIVE/COOPERATIVE HYPOTHESES
32	32	2.1	3.6	5.7

++An action is equivalent to one pattern description.

(d) ALTERNATIVE INTERPRETATIONS

TOTAL # OF DISTINCT COMPETITIVE INTERACTIONS	TOTAL # OF COMPETITIVE HYPOTHESES	AVG. # OF COMPETITIVE HYPOTHESES PER COMPETITIVE INTERACTION	MAXIMUM # OF ALTERNATIVE COMPETITIVE HYPOTHESES PER COMPETITIVE INTERACTION
59	69	1.2	3

Table VIII.4 Performance Statistics: Based on 19 Episode Types

for each competitive interaction. The maximum number of alternative hypotheses put forward for a competitive interaction was 3. In other words, while only one interpretation usually was hypothesized, sometimes conflicting interpretations for the same interaction were hypothesized.

In order to convey a sense of the amount of "machinery" *employed in* a single competitive hypothesis, consider the following actions:

[(101 THROW A1 PM BALL)(103 SWINGHIT B1 HP BALL)].

Two act-schemas of the existing nine were needed to make the inference that A1's THROWing the BALL was the act which enabled B1's subsequent HIT of the BALL. One CLS of the existing ten then made the hypothesis that a competitive relationship existed between the pitcher A1 and the batter B1. Of course, the activation of a CLS often involved the participation of as many as eight "predicates"; these predicates, in turn, required the involvement of a number of basic perceptual and inferential routines. Thus, in general, at the highest level the inference of a competitive (or cooperative) relationship required two act-schemas and one CLS, three knowledge sources out of nineteen.

VIII.1.3 Generalization and Verification

Before displaying the results of this run, let us review some aspects of the generalization and verification processes. First, both processes function in "real-time"; as the data is observed, episodes are generalized and verified (or eliminated). Since no external teacher is employed in the former process, the system must form its own classes. In particular, an episode class is formed on the basis of two (or more) episodes having the

same hypothesized competitive and cooperative interactions. Within an episode class, classes of individual interactions are formed. Such classes, and the episodes they compose, are evaluated on the basis of a confidence value. In this run, the confidence value on a hypothesis must reach 6 before it is accepted as truth. An episode is considered to be verified if the final competitive hypothesis and half of the other competitive hypotheses are verified.

The system's overall "box score" for this run is depicted in Tables VIII.5a and b. While the correct number of episode classes represented by the 19 episode types was 12, the system actually formed 31 classes and accepted (verified) 10 of them as correct (Table VIII.5a). Multiple and erroneous hypotheses caused a large number of classes to be formed initially. However, as the erroneous hypotheses were eliminated, the episode classes containing those hypotheses were also eliminated. Ten episode classes were finally verified, 9 of which were deemed correct in as much as they were based on a correct analysis of the observed activity (Figure VIII.1). The one incorrect episode class which should not have been verified was a class composed of flyouts; in these episodes, the system mistakenly hypothesized that a competitive timing relationship existed between the outfielder who was CATCHing the BALL and the batter who was reaching FIRSTBASE.

Some classes which one might expect to be formed were not; the system did not verify hypotheses which would have established a class of outfield singles, or those which would have established a class in which the pitcher

Episodes Formed = 31

Verified	10
Eliminated	7
Undecided	14

(a)

Episodes Verified = 10

Overgeneralied	2
Correctly Generalized	4
Undergeneralized	3

(b)

Episodes Which Should Not Have Been Verified

1

(c)

Table VIII.5 Box Score for Experiment #1: A Representative Run

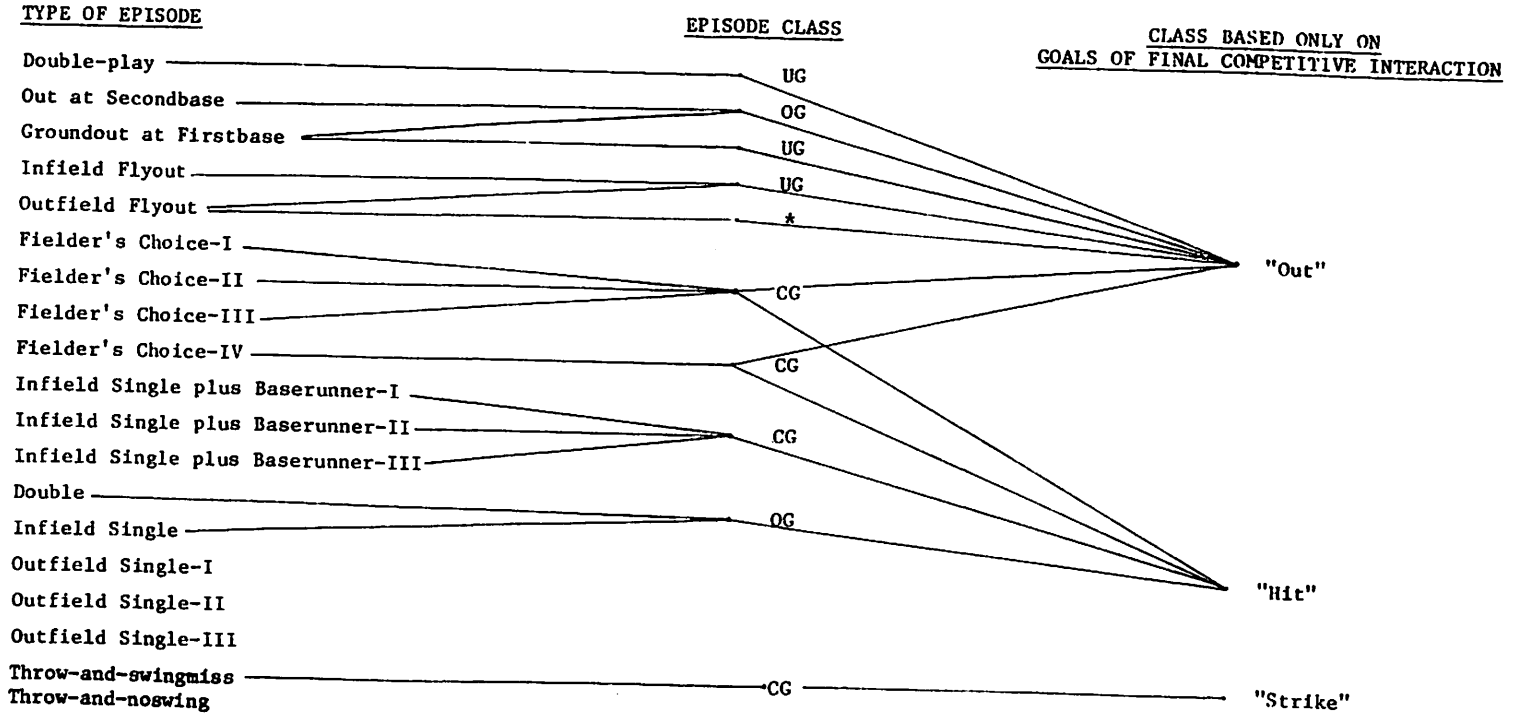


Figure VIII.1 Hierarchical Structure of Verified Classes

UG means undergeneralized, CG means correctly generalized, OG means overgeneralized, * indicates an episode class which should not have been verified since the analysis was incorrect.

throws the BALL and the batter does not swing his bat. Also, a large number (14) of episode classes were left "undecided". These classes were formed at the end of the run and really should have been merged with episode classes formed earlier. The problem was that a number of those classes were verified or eliminated before the features in the episodes were fully generalized. Once verified or eliminated, the features became frozen in their undergeneralized state and thus were no longer able to accomodate new episodes.

In Table VIII.5b we break down the verified episode classes along another interesting dimension: level of generality. In Chapter VI we described the rather "conservative" process by which features not hypothesized as relevant were generalized; variables were substituted for features which were constrained to match only those values which had already been observed. We will examine this policy in Experiment #2, and contrast its performance with an alternative, more ambitious generalization strategy. With respect to the current experiment, this generalization strategy resulted in a number of episode classes(5) being verified before they were sufficiently generalized. For example, 3 classes of groundouts were formed, 2 of which were verified (see Figure VIII.1), where each class covered only a portion of the possibly observable episodes of that type.

On the other hand, infield singles and doubles[50] were overgeneralized into the same class because of the order in which they were observed. That is, in the descriptions of the "infield single" and "outfield double"

provided to the system, there is little to which the system attends which discriminates between them; in both types of episodes a fielder throws the ball to the baseman, while the runner arrives at that base just before the opposing fielder at the base catches the ball.

At a higher level of abstraction the system formed three classes on the basis of only the final competitive goal of an episode (see Figure VIII.1). We have supplied the labels "hit", "strike", and "out"; they summarize the concepts acquired in the schemas. In the "fielder's choice episodes", there were two final competitive goals, one for the batter and one for the runner; either the batter failed or the runner failed with his goal of getting ON some base. Thus, this class of episodes participates in two classes -- hit and out -- at this higher level of abstraction.

There were many (93) classes of individual competitive interactions formed. However, the majority of these classes were identical, e.g., the competitive interaction between pitcher and the opposing batter appears in each of the episode classes. Thus, when duplicates are eliminated, the 36 classes of individual competitive interactions which are verified reduces to only 8 different classes. It is these classes, in their production rule representation, which are added to the set of Causal-Link Schemas and used to make specific inferences of goals and relationships in subsequently observed activity. Table VIII.6 lists the English equivalents of several of the learned rules.

1. If a player at HOMEPLATE HITs a BALL which was THROWn from the PITCHER'S-MOUND by a member of the opposing team,

   Then Hypothesize that the batter wanted to HIT the BALL and thus SUCCEEDed with his goal; the pitcher wanted to prevent the batter from performing that action and thus he FAILed with his goal.

2. If a player at HOMEPLATE SWINGs at a BALL and MISSes it, and an opposing player at the PITCHER'S-MOUND threw that BALL,

   Then Hypothesize that the batter did not want to miss the BALL and thus he FAILed with his goal; the pitcher wanted the batter to miss, thus the pitcher SUCCEEDed with his goal.

3. If a player arrives at FIRSTBASE or SECONDBASE before an opposing player at the base CATCHes the BALL,

   Then Hypothesize that the former player wanted to arrive at the base, and thus he SUCCEEDed with his goal; the latter player wanted to prevent that outcome, and thus FAILed with his goal.

4. If a player at FIRSTBASE or SECONDBASE CATCHes the BALL before an opposing player reaches the base, and who thereupon WALKS to his DUGOUT,

   Then Hypothesize that the latter player did not want to go to his DUGOUT and thus he FAILed with his goal; the former player wanted this outcome, and thus he SUCCEEDed with his goal.

5. If a player CATCHes a BALL which was HIT by an opposing player before it hits the ground,

   Then Hypothesize that the former player wanted to perform this action, and thus SUCCEEDed with his goal; the latter player did not want this outcome and thus he FAILed with his goal.

Table VIII.6 Learned Rules Expressed in English

VIII.2 Experiment #2: Alternative Generalization Strategies

In this section, we shall analyze the performance of two generalization (variable substitution) techniques: constrained data-directed generalization (CDDG) and unconstrained data-directed generalization (UDDG). Both techniques accommodate observed differences in the data by inserting a variable into the generalized pattern description to replace differing constants. In the former technique, that variable is constrained to subsequently match only pattern descriptions whose value for that feature is a member of the set of already observed values, e.g., a variable would be inserted into the generalized pattern description for the differing "location" features in (batter hits ball to centerfield) and (batter hits ball to rightfield) which would be constrained to match only instances of this pattern description in which the ball was hit to either centerfield or rightfield. The results presented in experiment #1 were obtained using this technique. Alternatively, in UDDG a variable is substituted which is allowed to match any value for that feature. The DDG systems described in Chapter VI adopy this latter strategy.

The choice of generalization strategy is important since our task differs in two crucial respects from most generalization situations reported in the literature: a) the data over which the system will generalize is not necessarily correct, and b) the data has not been partitioned into the correct classes by an external source. It is the interaction of these two "problems" which causes the trouble; we shall see that each technique

can cope with one of the problems but not with the other. In particular, we shall analyze the behavior of these two techniques with respect to their tendency to produce the correct level of generalization for episode classes and their role in the acceptance of incorrect hypotheses. We shall conclude that neither strategy produces completely desirable results and that additional knowledge needs to be employed in order to cope with the complex generalization situations arising in a real-world task such as baseball.

In the previous experiment the confidence value of a hypothesis was computed on the basis of predictions and consistency or inconsistency with previously acquired knowledge. In this experiment, however, we wanted to study only the effects of the alternative generalization strategies on the evaluation of a hypothesis. Thus, in this experiment we did not allow the system to use previously acquired knowledge to increase or decrease the confidence value of a hypothesis. Rather, the confidence value was based solely on the results of predictions. Since the level of generality of a hypothesis was directly reflected in its predictions, we were able to draw a clearer picture of the contribution of the alternative generalization methods to the evaluation process. In this setting then, a threshold value of 2 implies that 1 Type-I prediction would be needed to verify a hypothesis, while 2 would be needed for a threshold of 4, and 3 for a threshold of 6. Not unexpectedly, the elimination of the use of acquired knowledge in the verification process tends to worsen the system's overall performance, e.g., from Table VIII.5 we see that 9 episodes were correctly

verified when acquired knowledge was used in the evaluation process, but from Table VIII.8 we see that for the same threshold setting (6) only 5 episodes were verified when this information was not used. However, for the purposes of the following experiments this is not an important consideration.

Experiment A: The Behavior of CDDG and UDDG with respect to Generalization Level

In column 4 of Table VIII.7a we depict the results of running the system using CDDG with a threshold setting of 4. While 9 episode classes were verified, 3 of them were considered to be undergeneralized. For example, the class of "flyout" episodes was limited to matching only those episodes in which the ball was hit to either LEFTFIELD, RIGHTFIELD, or SHORTSTOP; this is clearly only a subset of the possible locations to which the BALL could be hit. The reason for this undergeneralization is actually quite simple. The system had previously observed several infield and outfield flyout episodes in which the BALL was HIT to only the locations listed above. Thus, the variable in the generalized episode description was constrained to match only those locations which had been observed so far. The confidence values for this class then reached threshold (two Type-I predictions were successfully matched) and the class was then accepted as correct. Since the features in the pattern description were frozen at that point, the system could not merge together any new flyout episodes in which the BALL were HIT to locations other than LEFTFIELD, RIGHTFIELD, or SHORTSTOP. A similar analysis holds for other undergeneralized classes.

	U D G	C D G	U D G	C D G	U D G	C D G	
	θ = 2		θ = 4		θ = 6		
Overgeneralized	1	1	1	2	1	1	(a)
Correctly Generalized	4	2	6	4	5	2	
Undergeneralized	5	8	0	3	0	2	
Episodes Which Should Not Have Been Verified	4	3	3	0	2	1	(b)
	(1)	(2)	(3)	(4)	(5)	(6)	

Table VIII.7 Performance of System: Experiments with UDDG and CDDG
θ represents the threshold setting.

We made a straightforward change to our generalization routines so that they would perform UDDG instead of CDDG. We then passed the same data through the system and Table VIII.7a also summarizes these results. For all threshold settings, UDDG formed fewer undergeneralized classes than did CDDG. Moreover, UDDG was not sensitive to the threshold setting at all once it reached the reasonable level of 4.

We can summarize the difference between the two approaches as follows:

(1) In order to be able to recognize a new episode, a system which employs CDDG needs to have seen an example of it already, while a system which employs UDDG needs only to have seen two examples before it creates a generalized template.

(2) Since UDDG can form classes "faster" than CDDG, it is reasonable that the former is less sensitive than the latter to threshold setting with regard to the level of generality of the classes formed.

At first blush, then, it would seem that UDDG is better at forming classes than its counterpart. However, the problems that arise with this technique will become evident in the next experiment when we discuss the role generalizations play in mistakenly accepting incorrect hypotheses as truths.

Experiment B: The Behavior of CDDG and UDDG with respect to Accepting Incorrect Hypotheses

The DDG systems described earlier in Chapter VI assume that the data on which they operate is correct; if it were not, their effectiveness would be severely handicapped in that such systems would tend to converge to the null (empty) generalization. For example, if one exemplar contains the crucial feature while another mistakenly does not, then the general rule which reflects the commonalities of both exemplars will not contain the

crucial feature. Unfortunately, our system cannot assume the correctness of the hypotheses. Therefore, in addition to building up correct generalizations, it must weed out the incorrect ones. It is just this interaction between the generalization strategy and the verification strategy which is troublesome.

In Experiment A, we saw that UDDG generalized "faster" than CDDG and thus tended to reach the correct level of class generalization sooner. However, UDDG's blind substitution of an unconstrained variable may also lead to overgeneralization. While the number of verified episode classes which were based on a correct analysis and which were overgeneralized is virtually the same for both generalization strategies (Table VIII.7a), Table VIII.7b tells a different story. That is, overgeneralizing a hypothesis -- and the corresponding predictions based on it -- which is incorrect tends to give that hypothesis more contexts in which it can be verified. Thus in Table VIII.7b we see that the UDDG strategy consistently accepted as truth more incorrect hypotheses than did the CDDG strategy.

The following example was taken from runs with the system in which UDDG was employed and where the threshold settings were 2, 4, or 6.

Outfield Single Episode Summary: (Batter HITS BALL to LEFTFIELD)
(Leftfielder CATCHES BALL after Batter arrives at FIRSTBASE)

Competitive Hypothesis: A timing relationship exists between the batter and the fielder; the batter was allowed to stay at FIRSTBASE because he arrived before the opposing player caught the BALL.

While the above hypothesis is incorrect, it is nonetheless reasonable since one could imagine a variant of baseball in which the above relationship were true. When the system subsequently sees another outfield single in which the ball is hit to CENTERFIELD, the system inserts into the generalized description a variable which will match any location. A corresponding variable insertion is performed on the Type-I prediction for this hypothesis which then states: if this hypothesis is true, the system should observe an event in which some fielder catches the ball before the batter reaches FIRSTBASE, and thus the batter is not permitted to perform ON FIRSTBASE. This prediction is readily matched in an infield or outfield flyout and thus the incorrect hypothesis is verified. The problem is the overgeneralization in the location feature of the incorrect hypothesis; infield locations FIRSTBASE and SECONDBASE do not behave the same as outfield locations LEFTFIELD, CENTERFIELD, etc. Thus, the location variable needs to be constrained in this case. A higher threshold setting reduces the system's tendency to accept incorrect hypotheses. Since "more time" is required to verify the incorrect one, a Type-II prediction may be triggered which would eliminate the incorrect hypothesis.

Since CDDG does not generalize as rapidly and blindly as UDDG, the range of possible situations in which an incorrect hypothesis can be verified is smaller, and thus the chances of observing such an event and matching a Type-I prediction are less. We ran the system using CDDG on the same data as was used above and found that the incorrect hypothesis was not accepted as truth. Of course, this technique will not eliminate the prob-

lem altogether; the system could have the misfortune of observing an unusual sequence of events which leads it to accept an incorrect hypothesis. However, a human in this situation might also make the same error.

VIII.3 Experiment #3: The Effects of Eliminating Some Knowledge About Games

In this section we shall describe two runs of the system in which different Causal-Link Schemas were excised from the system's initial knowledge base. The objective in this experiment is to explore the contribution of various major knowledge packets. In particular, we shall see that some knowledge does not seem to be critical and its elimination results in only local errors, while other knowledge is critical and its elimination results in catastrophic global errors. The conditions for both of these runs are the same as those in Experiment #1; the threshold is set to 6, and the number and type of episodes are as indicated in Table VIII.1.

In the first run of this experiment, the CLSs which were removed are the SUCCEED/FAIL and FAIL/SUCCEED versions of the competitive CLS "STATE-OF-DISTINGUISHED-OBJECT-COMPETITION" which hypothesizes a competitive relationship on the basis of the history of the ball[51] in conjunction with the actions of the two opposing players. From Table VIII.4a we see, that 8 of the 69 competitive inferences would be eliminated. While a number of eliminated hypotheses were incorrect, some were responsible for the

correct hypothesis of the goals and relationship in the final competitive interaction in the 'infield and outfield flyout' episodes.

Table VIII.8 presents the results of this run. The major difference between this run and the run in Experiment #1 is that one class of flyout episodes was not correctly verified. This is just what would be expected, since that class does depend on the presence of the CLS which was eliminated in this run. However, the class of flyouts based on an incorrect analysis was still able to be verified. That is, since the ORDER-OF-OCCURRENCE-COMPETITION CLS did make a hypothesis for the final competitive interaction in this class of episodes, and since the goals portion of that hypothesis was correct, these episodes could still be merged together.

In the second run of this experiment, both versions of the competitive timing relationship CLS "ORDER-OF-OCCURRENCE-COMPETITION" were eliminated. Again, the conditions for this run were the same as for the above run and for Experiment #1; the number and types of episodes were as listed in Table VIII.1, and the threshold was set at 6. We can see from Table VIII.4a that the number of inferences eliminated by the omission of these two CLSs comprised only 32% of the total number of competitive hypotheses. Nonetheless, their absence was critical since only 2 episode classes were verified (Table VIII.9). The episode class representing the batter SWINGing his bat and MISSing the BALL was verified as well as a class of flyouts; of course neither of these episodes involved the competitive timing relationship. The lack of episode classes correctly formed and verified is directly traceable to the system's ignorance of the timing relationship

Episodes Verified

Overgeneralized	1
Correctly Generalized	2
Undergeneralized	4

(a)

Episodes Which Should Not Have Been Verified

1

(b)

Table VIII.8 Performance of System When STATE-OF-DISTINGUISHED-OBJECT-COMPETITION CLSs Were Removed.

Episodes Verified

Overgeneralized	0
Correctly Generalized	1
Undergeneralized	1

(a)

Episodes Which Should Not Have Been Verified

0

(b)

Table VIII.9 Performance of System When ORDER-OF-OCCURRENCE-COMPETITION CLSs Were Removed

in this experiment; it is essential to a correct understanding of the basics of baseball.

VIII.4 Discussion

The results of the runs in Experiment #2 show that neither constrained nor unconstrained data-directed generalization produces completely satisfactory results -- each has its good points and each has its bad points. The moral to this story is that still more knowledge needs to be incorporated into the learning loop in order to accurately learn. For example, the system needed to have heuristics which would have enabled it to reason about the spatial characteristics of the field, e.g., once it had seen the BALL hit to CENTERFIELD and RIGHTFIELD, it should have inferred, based on geometric considerations of the playing field, that LEFTFIELD behaves just like CENTERFIELD and RIGHTFIELD. Lenat's system, AM, employs such projection rules in order to extend the range of its hypotheses. However, in loading a system with more and more knowledge in order to produce completely acceptable results, one may draw dangerously close to begging the question of learning. Quite possible this additional knowledge needs to reside in a teacher, or some other source external to the system.

Along similar lines, we should now return to two interrelated issues which were discussed in Chapter IV: did the system generate and search an interestingly large space of alternatives; and was the knowledge too highly tuned? Two pieces of evidence are significant in this regard:

1) the number of alternative hypotheses generated per hypothesized competitive interaction is, frankly speaking, low (1.2, see Table VIII,4d);

2) the collapse of the system when only one piece of knowledge was removed seems to indicate that the system was too inflexible (Experiment #3).

While we feel that the system did explore alternative interpretations using knowledge which was not tuned to baseball, we feel that the results of these experiments indicate that we were only moderately successful in this regard.

A more robust knowledge base and a more flexible hypothesis generation strategy needs to be employed in order to increase the total number of hypotheses and alternative hypotheses generated, and to increase the likelihood that one piece of knowledge would not be critical to the performance of the system. In particular, more knowledge about games needs to be incorporated into the system in the form of "smaller rules". That is, in their current formulation, each CLS examines 6-8 aspects of an interaction between two players, and requires that all of them be true in order for that CLS to trigger. Smaller rules, however, would test and be triggered by only 2 or 3 aspects. In addition, such rules could encode more subtle information than is presently incorporated in the CLSs, e.g., there could be rules which examine aspects of a player's behavior independent of that of any other player, and hypothesize the player's competitive or cooperative goal on the basis of this partial information. This analysis suggests that a hypothesis generation scheme in which numerous "cues" are accumulated needs to be explored (see [SHO76, LES77, HAN78]).

CHAPTER IX

CONCLUSIONS

In this concluding chapter, first we summarize the approach taken towards learning in this thesis. Next, we suggest some general design considerations in the use of multi-level architectures, followed by some proposals for further research suggested by our work. Finally, we speculate on the implications of our work.

IX.1 Learning = Interpretation + Generalization

In this thesis, we viewed the process of learning as one which can be decomposed into two other tasks: interpretation and generalization. "Interpretation" is the process of understanding some event (or events) in terms of a model of the domain. In computational terms, this process consists of moving from one level of description of the event to another level of description of the event, where the latter level includes new features which are meaningful to the domain. For example, our system moved from a description of isolated spatio-temporal events -- observations -- to a description of those events which included new features such as "goals-of-the-players" and "relationships-between-players". A general model of competitive action in games, provided initially to the system, facilitated this shift in the level of description. Verification is viewed as part of interpretation; some of the system's conjectures may be wrong and mechanisms must be employed to weed out the incorrect ones. "Generalization"

is the process of taking an interpretation which is specific to some event and permitting that description to be applicable in a wider range of contexts. This is accomplished by abstracting the relevant features from the interpretation. In effect, this process of abstraction allows similar events to be merged together into general classes of events.

Domain knowledge played a crucial role in both processes. It was used in the interpretation phase to deal with the absence of certain features (e.g., goals) in the observed descriptions; we saw that these features were critical to the correct functioning of the system. During generalization, domain knowledge was needed in order to highlight those features in the descriptions which were relevant to the task and the task domain -- inferring the goals of players in the competitive action game of baseball.

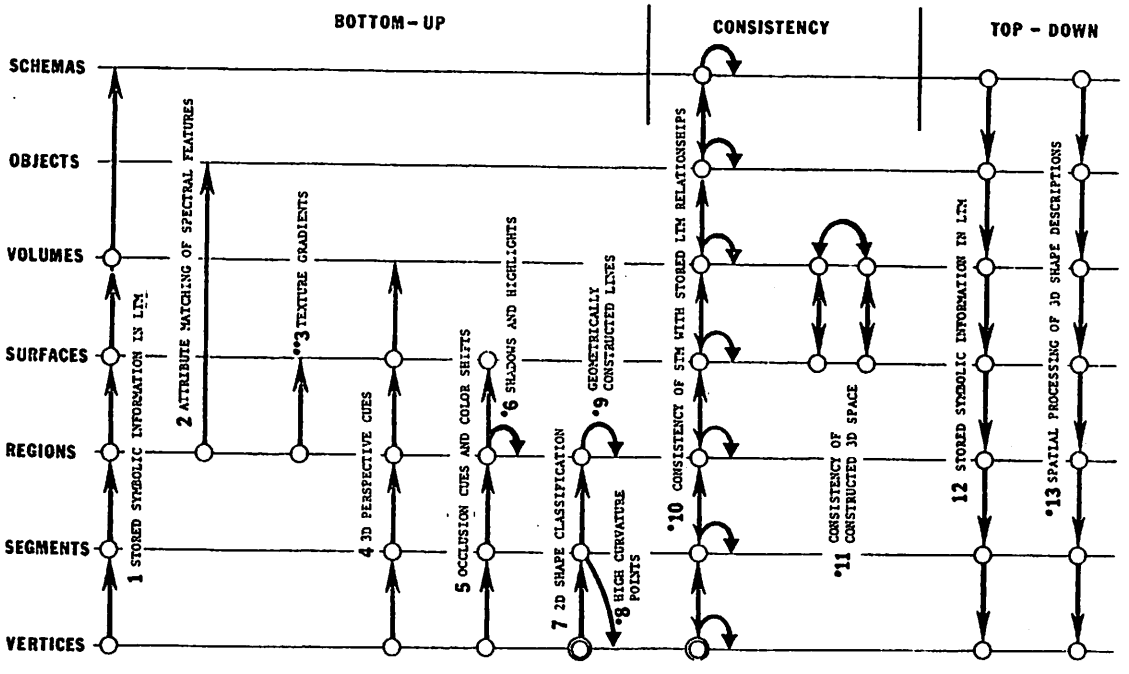
Finally, we explored how such domain knowledge could be incorporated into the learning loop -- the teacher, the observations, and the learning system. We argued that, typically, in rule induction systems which do not themselves employ domain knowledge, it is the teacher who provides that information in order to a) describe the observations at the appropriate level, b) partition the events in classes, and c) supply other training information. On the other hand, our system did not employ an explicit teacher. Rather we attempted to make explicit that information which a teacher would need to use in order to perform the above tasks, and provide it to the system itself. In this way, we felt that the contribution of domain knowledge to the learning process could be more clearly ascertained.

IX.2 The Multi-Level Architecture

The technique of decomposing the learning task into multiple levels of pattern description, knowledge, and processing facilitated both our conceptualization of the problem and the construction of a working *system*. The question is, however, what can be said in general about defining levels independently of a domain?

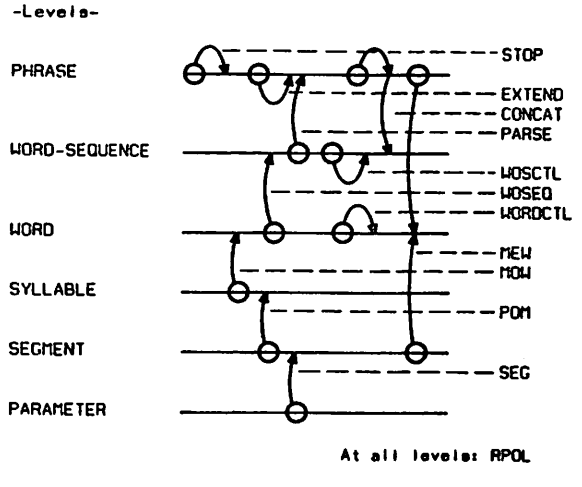
With only a handful of case studies available of systems which employ a multi-level architectural approach (e.g., [LES77, HAN78, NII77]), it is too early to draw concrete generalizations. However, since the levels in each of the above systems seem to correspond to "natural" structures in the domain, one might first look for analogies between these structures. For example, consider Figure IX.1a and b which depicts the levels structure in the scene interpretation system VISIONS and the speech understanding system HEARSAY-II, respectively. The relationship between a "word" and an "object", and a "syllable" and a "region" seem provocative and should be explored.

Two "meta-levels" of description can be identified in systems which deal with purposive behavior. In systems which deal with humans, as in baseball, or computer programs, as in problem solving, there are the non-intentional levels and the intentional (purposive) levels. We saw that levels 1-4 (Figure II.1) in our current system, which dealt with understanding the physical relationships between actions, would correspond to the non-intentional level of description. Only at levels 5 and 6 are players' goals finally introduced. We have also incorporated these two



The VISIONS Scene Analysis System (taken from [HAN78])

(a)



The HEARSAY-II Speech Understanding System (taken from [LES77])

(b)

Figure IX.1 Systems Which Employ a Multi-level Architecture

meta-levels into a system, currently under development, which is to learn LISP by reading annotated examples from the self-instructional text The Little LISPer [FRI74]. For example, non-intentional levels describe the structural relationships in the program (e.g., the test for an empty list is often the first test in a recursive program); the intentional levels describe the goals of the program and problem-solving method (e.g., the test for an empty list is used to stop the recursion from going into an infinite loop). These two meta-levels of descriptions can also be identified in the work of others involved in program understanding (e.g., [RIC76, GOL74, SUS73]).

IX.3 Areas for Further Investigation

A most interesting extension to this work would be the development and employment of an expanded knowledge base concerning concepts and structures in games. The additional hypotheses which would issue from such an augmented knowledge base would force us to reconsider the problems of control and search. A more sophisticated mechanism than is currently implemented would probably be needed to cope with the increased number of alternative interpretations put forth. As a side-effect, the identification of this additional knowledge about games might prove interesting to those studying the relationship of games to culture.

Another interesting area for exploration is error detection and error recovery. Currently, the system has no provisions for correcting an overgeneralization (e.g., merging the infield singles with the doubles,

Section VIII.1) or undoing the effects of accepting an incorrect hypothesis as truth (Section VIII.2). However, it is not obvious how such errors could even be detected by the system itself. For example, in the overgeneralization case mentioned above, there are no significant global consequences of that action. Even if such errors did have global effects and could be detected, the problems involved in backing-up and selectively undoing decisions are well-known. Error detection and recovery is a very difficult task.

The system currently processes the episodes as they appear. It does not modulate its processing based on an evaluation of the importance or complexity of an episode. A more sophisticated system ought to schedule the system's resources and tasks based on such an evaluation. What is required, then, is a focus-of-attention strategy coupled to an agenda of tasks; such mechanisms have been used with reportedly good results in various other systems (see, e.g., [LES77, BAR77, LEN76]).

IX.4 Final Remarks

The model proposed in this thesis attempts to demystify learning by reducing it to two processes which are already active areas of research. The general problem of "interpretation" is being explored in a whole range of problem areas: perceptual tasks [HAN78, LES77], story understanding [SCHA77, SCHM77b], medical diagnosis [SHO76], etc. The problem of "generalization" is also being investigated [VER77a, HAY77a, MIC77]. If our characterization is correct, then the process of learning has much in

common with other intelligence tasks and work in these related areas should bear directly on questions of learning. Simon and Lea [SIM74] make a similar argument in their demonstration of how learning can fit into the hypothesize-and-test paradigm which was initially developed for problem solving tasks. However, there are only a few research efforts which attempt to integrate interpretation with generalization (e.g., [WIN70, EGA68]). Also, problems which are unique to learning such as the generality of the initial knowledge base, the use of acquired knowledge, and the interaction between generalization and verification must be considered. Nonetheless we feel that the perspective provided in this thesis suggests areas in which fruitful research can be carried out.

FOOTNOTES

[1]The discussions in this thesis assume only a passing familiarity with the game of baseball. The reader unacquainted with this game, but familiar with other action games should be able, from the context of the discussions, to understand the points at issue -- afterall, this is one of the points of the thesis! A brief summary of the highlights of the game is provided in an appendix.

[2]At the outset, let us be clear about what we mean by features. Generally speaking, we shall use this term to indicate all levels of features; from simple perceptual features such as action, location, to higher level features such as goal and causal relationship, to composite ones such as plan. We shall qualify the general term "features" only when we wish to draw attention to a specific type or level of feature.

[3]The complimentary hypothesis might also be possible: a player who enabled an opponent's action might actually have wanted to have that outcome occur, while the opponent had not wanted to perform the observed act. The system is provided with both rules; Chapter IV discusses how the system determines the applicability of the conflicting rules.

[4]The notation which we shall use in the figures throughout this thesis is to be interpreted as follows: variables in pattern descriptions are signified by a ? or $ prefixing the variable's name, while constants are not prefixed. Values for the variables are supplied by the system's pattern matches; a variable prefixed by a ? indicates that the pattern matcher should fill in some value for the variable, while a $ indicates that the pattern matcher should simply retrieve the value already assigned to the variable. Moreover, the binding (assignment) of a value to a variable may be made conditional upon the evaluation of a predicate; this is indicated by a parenthesized expression following the ?. The usage of constants and variables follows that of PLANNER-like languages [SUS71, BOB74].

[5]This delightful quote was shown to me by Michael Arbib. A long story goes along with it, but I will spare the reader the details.

[6]The names which the system uses internally are simply strings of symbols, G0001, G0002, etc. In this thesis, we shall replace such names with meaningful ones, such as infield single. We do not feel that this relabeling misrepresents the system's behavior.

[7]With respect to the choice of "schemata" or "schemas" as the correct English plural of "schema", we quote Arbib [ARB77]

> Most English dictionaries offer the Greek plural schemata for the word schema, but we prefer the English plural schemas. Fowler's 'Modern English Usage' states: 'Of most words in fairly common use that have a Latin as well as or instead of an English plural the correct Latin form is given in the word's alphabetical place...There is a tendency to abandon the Latin plurals, and when one is really in doubt which to use the English form should be given the preference.' (The cited comments refer to Latinized-Greek as well.) This tendency to abandon classical plurals is part of the pattern of historical change of English. For example, the 19th century had already seen dogmas achieve parity with dogmata, and in current usage only the English form appears unaffected. Amongst authors who share our preference for schemas, we may cite P.H. Lindsay and D.A. Norman: 'Human Information Processing', 2nd Edition, Academic Press (1977).

[8]A production rule is composed of two parts: a condition and an action. If the condition evaluates to true, then the action is taken.

[9]There are 9 players on each time and a number (6) of scoreboard markers (see Appendix).

[10]For the sake of simplicity we have limited purposive descriptions to the area of human behavior. Clearly, the same arguments hold for all animate entities and at least some inanimate objects (see ROS43).

[11]Since researchers on games are always observing and learning new games, it might be interesting to interrogate them as to their knowledge and learning strategy.

[12]For the most part this analysis holds for the degenerate case when there is only 1 player on a team.

[13]Ignore for a moment the distinction between the batter SWINGing the BAT and MISSing the BALL, and the batter not SWINGing the BAT at all.

[14]We number the players and their corresponding actions in the above way in order to be consistent with the numbering of the players and their actions in the previous section. The PLAYER2s and their ACT2s will always be the actions "enabled" by the PLAYER1s and ACT1s. Here "enabled" is meant to cover all types of competitive and cooperative enablement.

[15]In the current running version of the program, probabilities or confidence factors are not assigned to the hypotheses during Hypothesis Generation, though a crude confidence measure is employed during Hypothesis Evaluation. While this simplification was sufficient for our current purposes, an expanded version of this system might well need such probabilities at this level of computation.

[16]In order to thoroughly analyze the outcome structure of a cooperative interaction, the system would need to take into account the subsequent competitive interactions. We shall discuss this issue in Section IV.3.5.

[17]We show how this case is handled in Chapter VI.

[18]The following discussion draws on the published work of Schmidt and Sridharan ([SCHM76a, SCHM76b, SCHM77b]) and the many personal discussions with them.

[19]The same argument seems true for board games such as chess and checkers. The great deal of anthropomorphizing which goes on in the description of the play and the strategies in such games does not seem accidental.

[20]We are, of course, ignoring the personal goals -- the non-game goals -- of the players, e.g., fame, glory, money, love.

[21]In Chapter V, we shall see how the feature FINAL-COMPETITIVE-INTERACTION is added to the appropriate pattern descriptions in an episode.

[22]This discussion was particulary influenced by the work of Schmidt and Sridharan [SCHM76b].

[23]The ideas presented in this section are only suggestions and have <u>not</u> been implemented in our system.

[24]The suggestions in this section also have not been implemented.

[25]Eight CLSs were described in this chapter; two more will be described in Section V.3.

[26]The threshold setting for accepting hypotheses as truths, the order of observation of the data, the "representativeness" of the data all contribute to this result (see Chapter VIII).

[27]The situation is much like that of Simon's ant [SIM69]; while the ant seems to be exhibiting complex behavior in traversing a rugged terrain back to its nest, actually, simple mechanisms in the ant together with rich environmental constraints can actually account for the same behavior.

[28]For increased readability, all register names end with an asterisk.

[29]If the rules are run in the forward direction, then the LHSs are matched. However, if the rules are run in the backwards direction, then the right-hand sides are matched against the data base first.

[30]From Figure V.3 we see that the competitive CLSs are applied in one network (4), and the cooperative CLSs are applied in another (5). Thus it would be more accurate to say that there is one recognize-activate cycle per action for the analysis of competitive interactions and one for the analysis of cooperative interactions.

[31]The sequence of events depicted in Figure V.6 is termed a "called-strike" in baseball.

[32]In collaboration with J. Lowrance [LOW75], we have taken some steps in this direction. For example, the global nature of the registers is especially troublesome; a binding scheme similar to LISP's would remedy this situation.

[33]The discussion in this section follows closely that of Salmon's [SAL67].

[34]We are not claiming that there is a system which relies solely on R_2 for rule induction. As we shall see, actual schemes embellish R_2. The above simplification allows us to analyze the basic problems which arise in rule induction and thus illustrate why those embellishments are necessary.

[35]We have purposely chosen not to represent some features which are evident in Scene 1 and Scene 2 in the symbolic descriptions of those scenes, e.g., while LARGE and WHITE are features of the square in Scene 1, they are not included in the symbolic description of Scene 1. These omissions were meant to symbolize the fact that DDG works on the features which are provided; it is the responsibility of a trainer or an earlier phase of processing (see Chapter IV) to guarantee that the appropriate set of features are included in the descriptions.

[36]The terminology "variabilizing" and "dropping" appears in [LAR77].

[37]Michalski calls the range of parameters specifiable by a user "optimality criteria".

[38]Hayes-Roth [HAY76] and Michalski [LAR77] acknowledge the possibility of their systems not discovering the actual underlying rule even given these constraints, since heuristic knowledge could have over-ridden their otherwise exhaustive and effective algorithms. That is, an LCG which seemed to be less valuable than some others may have been eliminated too early; subsequent exemplars would have shown that LCG to be the correct one.

[39]We have used the term 'least common generalization', while Vere [VER77b] has called the same concept 'maximal common generalization'.

[40]Winston actually called it a "difference ordering".

[41]The notation

```
?(LOCATION (MEMBER $LOCATION '(SECONDBASE THIRDBASE)))
```

is interpreted as follows: the pattern matcher first assigns a location to the variable LOCATION (?LOCATION), and then it tests to see if the location ($LOCATION) is a member of the set of specified locations (SECONDBASE THIRDBASE). If true, then the match is successful; if false, then the matcher tries to find another location which satisfies the conditions.

[42]While Vere's formulation actually permits multiple equivalent generalizations to be generated, he does employ a similar constraint in the apparent hope of decreasing the likelihood of this eventuality.

[43]We remind the reader that such terms as "out", "infield groundout", etc., are labels which we employ and are not provided to the system a priori or generated by the system.

[44]A "no-hitter", in which the pitcher (and his team) is always successful, is meaningful only because the normal state of affairs involves a mixture of successes and failures.

[45]The names **P1, **H1, **H3, etc., are generated internally by the system for bookkeeping purposes.

[46]This discussion is based on the version-1-CLS: the FAILure of player A enables the SUCCESS of player B. A similar analysis would go through for version-2-CLSs: the SUCCESS of player A forced the FAILure of player B.

[47]Recall from our discussion in Chapter IV, that B1's action would be considered ACT2 and A3's action ACT1. Thus, F/S means A3 FAILed and B1 SUCCEEDed.

[48]Of course, there are other conditions which must be true; see Section IV.3.2.2 for details.

[49]A Type-I prediction predicts the complementary outcome of a competitive interaction: a SUCCEED/FAIL outcome structure predicts a FAIL/SUCCEED outcome structure and vice versa. A Type-II prediction predicts what events should not be observed if the rule in question is correct. A match of a Type-II prediction serves to eliminate the hypothesis, while a match of a Type-I prediction increases the confidence value of the hypothesis on which that prediction is based.

[50]We remind the reader that in the "double" episode, the batter hits the ball to the outfield, and the opposing fielder there catches the ball and throws it to his teammate at secondbase; it arrives at secondbase shortly after the batter reaches that base.

[51]We remind the reader that the "history of the ball" includes whether or not it has touched the ground after being hit by the batter.

[52]This is a simplification, and is, strictly speaking incorrect. Whereas the fielder at firstbase needs only to stand on firstbase in possession of the ball in order to prevent the batter from remaining on firstbase, fielders at secondbase and thirdbase must actually touch the batter with hand holding the ball before he arrives at the base. This level of detail is not included in the descriptions of activity presented to our system.

BIBLIOGRAPHY

ARB72 Arbib, M. (1972) The Metaphorical Brain. John Wiley and Sons, New York

AVE71a Avedon, E.M. and B. Sutton-Smith (1971) The Study of Games. John Wiley and Sons, New York

AVE71b Avedon, E.M. (1971) The Structural Elements of Games. In The Study of Games Avedon and Sutton-Smith (Eds.), John Wiley and Sons, New York

BAC74 Bach, E. (1974) Syntactic Theory. Holt, Rinehart and Winston, Inc., New York

BAR77 Barstow, D. (1977) A Knowledge-Based System for Automatic Program Construction. Proc. of the Fifth International Joint Conference on Artificial Intelligence, Cambridge, Mass.

BOB74 Bobrow, D.G. and B. Raphael (1974) New Programming Languages for Artificial Intelligence Research. ACM Computing Surveys, VI, pp. 155-174, September

BOB75 Bobrow, D. (1975) Dimensions of Representation. In Representation and Understanding, Bobrow and Collins (Eds.), Academic Press, New York

BOB77a Bobrow, D.G. and T. Winograd (1977) An Overview of KRL-0, A Knowledge Representation Language. Cognitive Science, Vol. 1, No. 1

BOB77b Bobrow, D.G. (1977) A Panel on Knowledge Representation. Proc. of the Fifth International Joint Conference on Artificial Intelligence, Cambridge, Mass.

BRO75 Brown, J.S. (1975) Multiple Representations of Knowledge for Tutorial Reasoning. In Representation and Meaning, Bobrow and Collins (Eds.), Academic Press, New York

BUC73 Buchanan, B.G., E.A. Feigenbaum and J. Lederberg (1973) A Heuristic Programming Study of Theory Formation in Science. Proc. of the Second International Joint Conference on Artificial Intelligence, London

BUC77 Buchanan, B.G. and T. Mitchell (1977) Model-Directed Learning of Production Rules. In Proceedings of the Workshop on Pattern-Directed Inference Systems, Hayes-Roth and Waterman (Eds.), Hawaii, in press

CHA75 Chafe, W.L. (1975) Some Thoughts on Schemata. Proceedings of a Workshop on Theoretical Issues in Natural Language Processing, Cambridge, Mass.

CHA72 Charniak, E. (1972) Towards a Model of Children's Story Comprehension. M.I.T. A.I. Lab Memo TR-266, Cambridge, Mass.

DAT76 Date, C.J. (1976) An Introduction to Database Systems. Addison-Wesley Publishing Co., Reading, Mass.

DAV75 Davis, R. and J. King (1975) An Overview of Production Systems. Stanford University, Dept. of Computer Science Memo STAN-CS-75-524, AI MEMO AIM-271, Stanford, Calif.

DAV76 Davis, R. (1976) Applications of Meta-Level Knowledge to the Construction, Maintenance, and Use of Large Knowledge Bases. Stanford University, A.I. Lab Memo 283, Stanford, Calif.

DEN66 Dennis, J.B. and E.C. Van Horn (1966) Programming Semantics for Multiprogrammed Computations. CACM 9:3, March

DUD77 Duda, R., P. Hart, N.Nilsson and G. Sutherland (1977) Semantic Network Representations in Rule-Based Inference Systems. In Proceedings of the Workshop on Pattern-Directed Inference Systems, Hayes-Roth and Waterman (Eds.), Hawaii, in press

EGA74 Egan, D.E. and J.G. Greeno (1974) Theory of Rule Induction: Knowledge Acquired in Concept Learning, Serial Pattern Learning, and Problem Solving. In Knowledge and Cognition, L. Gregg (Ed.), Lawrence Erlbaum Associates, Potomac, Maryland

EIN18 Einstein, A. (1918) Principles of Research. Reprinted in Ideas and Opinions, Dell Publishing Co., New York, p. 221

EIN36 Einstein, A. (1936) Physics and Reality. Reprinted in Ideas and Opinions, Dell Publishing Co., New York, p. 299

EVA68 Evans, T.G. (1968) A Program for the Solution of Geometric-Analogy Intelligence Test Questions. Ph.D. Thesis, M.I.T., reprinted in Semantic Information Processing, M. Minsky (Ed.), MIT Press, Cambridge, Mass.

FEI77 Feigenbaum, E. (1977) The Art of Artificial Intelligence: Themes and Case Studies of Knowledge Engineering. Proc. of the Fifth International Joint Conference on Artificial Intelligence, Cambridge, Mass.

FIK72 Fikes, R.E., P.E. Hart and N. Nilsson (1972) Learning and Executing Generalized Robot Plans. Artificial Intelligence 3, pp. 251-288

FRI74 Friedman D. (1974) The Little LISPer. Science Research Associates, Palo Alto, Calif.

GOL74 Goldstein, I. (1974) Understanding Simple Picture Programs. Ph.D. Thesis, M.I.T. A.I. Lab Memo TR-294, Cambridge, Mass.

HAN76 Hanson, A. and E. Riseman (1976) A Progress Report on VISIONS: Representation and Control in the Construction of Visual Models. University of Massachusetts, Dept. of Computer Science Report 76-9, Amherst, Mass.

HAN78 Hanson, A. and E. Riseman (1978) VISIONS: A Computer System for Interpreting Scenes. In Computer Vision Systems, Hanson and Riseman (Eds.), Academic Press, New York

HAY77d Hayes, P.J. (1977) In Defense of Logic. Proc. of the Fifth International Joint Conference on Artificial Intelligence, Cambridge, Mass.

HAY75 Hayes-Roth, F. (1975) Collected Papers on the Learning and Recognition of Structured Patterns. Carnegie-Mellon University, Dept. of Computer Science Report, Pittsburgh, Pa.

HAY76 Hayes-Roth, F. and J. McDermott (1976) Knowledge Acquisition from Structural Descriptions. Carnegie-Mellon University, Dept. of Computer Science Report, Pittsburgh, Pa.

HAY77a Hayes-Roth, F. (1977) The Role of Partial and Best Matches in Knowledge Systems. In Proceedings of the Workshop on Pattern-Directed Inference Systems, Hayes-Roth and Waterman (Eds.), Hawaii, in press

HAY77b Hayes-Roth, F. and J. McDermott (1977) Knowledge Acquisition from Structural Descriptions. Proc. of the Fifth International Joint Conference on Artificial Intelligence, Cambridge, Mass.

HAY77c Hayes-Roth, F. (1977) Uniform Representations of Structured Patterns and an Algorithm for the Induction of Contingency-Response Rules. Information and Control, 33, pp. 87-116

HEN73 Hendrix, G. (1973) Modelling Simultaneous Actions and Continuous Processes. Artificial Intelligence, 4, pp. 145-180

HEN76 Hendrix, G. (1976) Representation of Semantic Knowledge and Semantic Aspects of Translation. In Speech Understanding Research: Final Report, Project 4762, D. Walker (Ed.), A.I. Center, Stanford Research Institute, Menlo Park, Calif.

HEW72 Hewitt, C. (1972) Description and Theoretical Analysis (Using Schemata) of PLANNER: A Language for Proving Theorems and Manipulating Models in a Robot. M.I.T. A.I. Lab Memo TR-258, Cambridge, Mass.

JOR68 Jordan, M. (1968) New Shapes of Reality, Aspects of A.N. Whitehead's Philosophy. George Allen and Unwin Ltd., London, pp. 154-155

JOW49 Jowett, B. (1949) Plato's Meno, translated by B. Jowett. The Liberal Arts Press Inc., Indianapolis, Indiana

KON75 Konolige, K. (1975) ALISP Users Manual. University Computing Center, University of Massachusetts, Amherst, Mass.

KUL76 Kulikowski, C., S. Weiss, M. Trigoboff and A. Safir (1976) Clinical Consultation and the Representation of Disease Processes: Some AI Approaches. Proceedings of the Conference on Artificial Intelligence and Simulation of Behavior, Edinburgh

LAR77 Larson, J. and R. Michalski (1977) Inductive Inference of VL Decision Rules. SIGART Newsletter, No. 63, June

LEN76 Lenat, D.B. (1976) AM: An Artificial Intelligence Approach to Discovery in Mathematics as Heuristic Search. Stanford University, A.I. Lab. Report AIM-286 Stanford, Calif.

LEN77 Lenat, D.B. (1977) Automated Theory Formation in Mathematics. Proc. of the Fifth International Joint Conference on Artificial Intelligence, Cambridge, Mass.

LES77 Lesser, V. and L.D. Erman (1977) A Retrospective View of the HEARSAY-II Architecture. Proc. of the Fifth International Joint Conference on Artificial Intelligence, Cambridge, Mass.

LOW75 Lowrance, J. (1975) Design and Implementation of an ATN Generator. Unpublished report

McC59 McCarthy, J. (1959) Programs With Common Sense. Reprinted in Semantic Information Processing, M. Minsky (Ed.) MIT Press, Cambridge, Mass.

McC65 McCarthy, J., et al. (1965) LISP 1.5 Programmer's Manual. MIT Press, Cambridge, Mass.

McC69 McCarthy, J. and P.J. Hayes (1969) Some Philosophical Problems from the Standpoint of Artificial Intelligence. In Machine Intelligence 4, D. Michie (Ed.), Edinburgh University Press, Edinburgh

McD77 McDermott, J. and C. Forgy (1977) Production System Conflict Resolution Strategies. In Proceedings of the Workshop on Pattern-Directed Inference Systems, Hayes-Roth and Waterman (Eds.), Hawaii, in press

MIC77 Michalski, R. (1977) Toward Computer-Aided Induction. Carnegie-Mellon University, Dept. of Computer Science Technical Report, Pittsburgh, Pa.

MIN63 Minsky, M. (1963) Steps Toward Artificial Intelligence. In Computers and Thought, Feigenbaum and Feldman (Eds.), McGraw-Hill Book Co., New York

MIN72 Minsky, M. and S. Papert (1972) Artificial Intelligence: Progress Report. M.I.T. A.I. Lab Memo 252, Cambridge, Mass.

MIN75 Minsky, M. (1975) A Framework for Representing Knowledge. In The Psychology of Computer Vision, P. Winston (Ed.), McGraw-Hill Publishing Co., New York

MIT77 Mitchell, T. (1977) Version Spaces: A Candidate Elimination Approach to Rule Learning. Proc. of the Fifth International Joint Conference on Artificial Intelligence, Cambridge, Mass.

MOO73 Moore, J. and A. Newell (1973) How Can MERLIN Understand? In Knowledge and Cognition, L. Gregg (Ed.), Lawrence Erlbaum Associates, Potomac, Maryland

NEW59 Newell, A., J. Shaw and H. Simon (1959) Report on a General Problem-Solving Program. Proc. of the International Conference on Information Processing, UNESCO House, Paris

NEW73 Newell, A. (1973) Production Systems: Models of Control Structures. In Visual Information Processing, W.C. Chase (Ed.), Academic Press, New York

NEW76 Newell, A. and H. Simon (1976) Computer Science as an Empirical Inquiry: Symbols and Search. CACM 19:3, pp. 113-126

NII77 Nii, P. and E.A. Feigenbaum (1977) Rule Based Understanding of Signals. In Proceedings of the Workshop on Pattern-Directed Inference Systems, Hayes-Roth and Waterman (Eds.), Hawaii, in press

NIL77 Nilsson, N. (1977) Lecture Notes from a Seminar on Expert Systems. University of Massachusetts, Dept. of Computer Science, Amherst, Mass.

PIA65 Piaget, J. (1965) The Moral Judgement of the Child. The Free Press, New York

PLO70 Plotkin, G.D. (1970) A Note on Inductive Generalization. In Machine Intelligence 5, Meltzer and Michie (Eds.), American Elsevier, New York

PLO71 Plotkin, G.D. (1971) A Further Note on Inductive Generalization. In Machine Intelligence 6, Meltzer and Michie (Eds.), American Elsevier, New York

POP59 Popper, K. (1959) The Logic of Scientific Discovery. Basic Books Inc., New York

POP77 Pople, H.E. (1977) The Formation of Composite Hypotheses in Diagnostic Problem Solving: An Exercise in Synthetic Reasoning. Proc. of the Fifth International Joint Conference on Artificial Intelligence, Cambridge, Mass.

QUI69 Quillian, M.R. (1969) The Teachable Language Comprehender. CACM, 12, pp. 459-476

REY70 Reynolds, J.C. (1970) Transformational Systems and the Algebraic Structure of Atomic Formulas. In Machine Intelligence 5, Meltzer and Michie (Eds.) American Elsevier, New York

RIC76 Rich, C. and H. Shobe (1976) Initial Report on a LISP Programmer's Apprentice. M.I.T. A.I. Lab Memo TR-354, Cambridge, Mass.

RIE73 Rieger, C. (1973) The Common-Sense Algorithm as a Basis for Computer Models of Human Memory. The Proceedings of the Conference on Theoretical Issues in Natural Language Processing, Cambridge, Mass.

RIS77 Riseman, E. and M.A. Arbib (1977) Computational Techniques in the Visual Segmentation of Static Scenes. Computer Graphics and Image Processing, pp. 221-276

ROB65 Robinson, J.H. (1965) A Machine Oriented Logic Based on the Resolution Principle. JACM 12, 1

ROS43 Rosenblueth, A., N. Wiener and J. Bigelow (1943) Behavior, Purpose and Teleology. Philosophy of Science, 10, 18

RUM75 Rumelhart, D.E. (1975) Notes on a Schema for Stories. In Representation and Understanding, Bobrow and Collins (Eds.), Academic Press, New York

RUT76 Ruth, G. (1976) Intelligent Program Analysis. Artificial Intelligence, 7, p. 65-85

SAL67 Salmon, W.C. (1967) The Foundations of Scientific Inference. University of Pittsburgh Press, Pittsburgh, Pa.

SCHA73a Schank, R.C. (1973) Conceptualizations Underlying Natural Languages. In Computer Models of Thought and Language, Schank and Colby (Eds.), Freeman Press, New York

SCHA73b Schank, R.C. (1973) Understanding Paragraphs. Instituto per gli Studi Semantici e Cognitivi, Technical Report 5, Castognola, Switzerland

SCHA75 Schank, R.C. and R.P. Abelson (1975) Scripts, Plans and Knowledge. Advanced Papers of the Fourth International Joint Conference on Artificial Intelligence, Tbilisi, U.S.S.R.

SCHA77 Schank, R.C. and R.P. Abelson (1977) Goals, Plans, Scripts and Understanding: An Enquiry into Human Knowledge Structures. Lawrence Erlbaum Press, N.J.

SCHM76a Schmidt, C.F. (1976) Understanding Human Action: Recognizing the Plans and Motives of Other Persons. In Cognition and Social Behavior, Carrol and Payne (Eds.), Lawrence Erlbaum Press, N.J.

SCHM76b Schmidt, C.F., M.S. Sridharan and J.L. Goodson (1976) Recognizing Plans and Sumarizing Actions. Proceedings of the Conference on Artificial Intelligence and Simulation of Behavior, Edinburgh

SCHM77a Schmidt, C.F. (1977) personal communication

SCHM77b Schmidt, C.F. and N.S. Sridharan (1977) Plan Recognition Using a Hypothesize and Revise Paradigm: An Example. Proc. of the Fifth International Joint Conference on Artificial Intelligence, Cambridge, Mass.

SEL59 Selfridge, O.G. (1959) Pandemonium: A Paradigm for Learning. Symposium on the Mechanization of Thought Processes, H.M. Stationery Office, London

SHO76 Shortliffe, E. (1976) Computer-Based Medical Consultations: MYCIN. Elsevier Scientific Publishing Co., New York

SIM69 Simon, H.A. (1969) The Sciences of the Artificial. MIT Press, Cambridge, Mass.

SIM74 Simon, H.A. and G.L. Lea (1974) Problem Solving and Rule Induction: A Unified View. In Knowledge and Cognition, L. Gregg (Ed.), Lawrence Erlbaum Associates, Potomac, Maryland

SOL66 Solomonoff, R.J. (1966) Some Recent Work in Artificial Intelligence. Proceedings of the IEEE, Vol. 54, No. 12

SOL75 Soloway, E.M. and E. Riseman (1975) Common-sense Theory Formation: Towards Understanding Baseball. University of Massachusetts, Dept. of Computer Science Technical Report 75C-5, Amherst, Mass.

SOL76 Soloway, E.M. and E. Riseman (1976) Mechanizing the Common-sense Inference of Rule Which Direct Behavior. Proceedings of the Conference on Artificial Intelligence and the Simulation of Behaviour, Edinburgh

SOL77a Soloway, E.M. and E. Riseman (1977) Knowledge-Directed Learning. SIGART Newsletter, No. 63, June

SOL77b Soloway, E.M. and E. Riseman (1977) Levels of Pattern Description in Learning. Proc. of the Fifth International Joint Conference on Artificial Intelligence, Cambridge, Mass.

SRI76 Sridharan, N.S. (1976) The Frame and Focus Problems in AI: Discussion in Relation to the Believer System. Proceedings of the Conference on Artificial Intelligence and the Simulation of Behavior, Edinburgh

STA76 Stallman, R. and G.J. Sussman (1976) Forward Reasoning and Dependency-Directed Backtracking in a System for Computer-Aided Circuit Analysis. M.I.T. A.I. Lab Memo 380, Cambridge, Mass.

SUS71 Sussman, G.J., et al. (1971) Micro-PLANNER Reference Manual. M.I.T. A.I. Lab Memo 203A, Cambridge, Mass.

SUS73 Sussman, G.J. (1973) A Computational Model of Skill Acquisition. M.I.T. Dept. of Mathematics, Ph.D. Thesis, M.I.T. A.I. Lab Memo TR-297, Cambridge, Mass.

SUS77 Sussman, G.J. (1977) Electrical Design: A Problem for Artificial Intelligence Research. Proc. of the Fifth International Joint Conference on Artificial Intelligence, Cambridge, Mass.

VER75 Vere, S.A. (1975) Induction of Concepts in the Predicate Calculus. Advanced Papers of the Fourth International Joint Conference on Artificial Intelligence, Tbilisi, U.S.S.R.

VER77a Vere, S.A. (1977) Inductive Learning of Relational Productions. In Proceedings of the Workshop on Pattern-Directed Inference Systems, Hayes-Roth and Waterman (Eds.), Hawaii, in press

VER77b Vere, S.A. (1977) Induction of Relational Productions in the Presence of Background Information. Proc. of the Fifth International Joint Conference on Artificial Intelligence, Cambridge, Mass.

WAL77 Waldinger, R.J. (1977) Achieving Several Goals Simultaneously. In Machine Intelligence 8: Machine Representations of Knowledge, Elcock and Michie (Eds.), John Wiley and Sons, New York, to appear

WAT70 Waterman, D.A. (1970) Generalization Techniques for Automating the Learning of Heuristics. Artificial Intelligence 1, pp. 121-170

WEI77 Weiss, S.M., C.A. Kulikowski and A. Safir (1977) A Model-Based Consultation System for the Long-Term Management of Glaucoma. Proc. of the Fifth International Joint Conference on Artificial Intelligence, Cambridge, Mass.

WIL74 Wilks, Y. (1974) Natural Language Understanding Systems Within the AI Paradigm: A Survey and Some Comparisons. Stanford University, A.I. Lab Memo AIM-237, Stanford, Calif.

WIL75 Wilks, Y. (1975) Seven Theses on Artificial Intelligence and Natural Language. Instituto per gli Studi Semantici e Cognitivi, Report No. 17, Castognola, Switzerland

WIN75 Winograd, T. (1975) Frame Representations and the Declarative-Procedural Controversy. In Representation and Understanding, Bobrow and Collins (Eds.), Academic Press, New York

WIN70 Winston, P.H. (1970) Learning Structural Descriptions from Examples. M.I.T. Project MAC Report 76, Cambridge, Mass.

WOO73 Woods, W.A. (1973) An Experimental Parsing System for Transition Network Grammars. In Natural Language Processing, R.Rustin (Ed.), Algorithmic Press, New York

WOO75 Woods, W.A. (1975) What's in a Link: Foundations for Semantic Networks. In Representation and Understanding, Bobrow and Collins (Eds.) Academic Press, New York

ZIS77 Zisman, M.D. (1977) Use of Production Systems for Modeling Asynchronous, Concurrent Processes. In Proceedings of the Workshop on Pattern-Directed Inference Systems, Hayes-Roth and Waterman (Eds.), Hawaii, in press

APPENDIX A

A PRIMER ON THE GAME OF BASEBALL

Baseball is a competitive action game played between two teams, each composed of at least nine players. The game is played with a ball and a stick, approximately 30 inches in length, which is referred to as a bat; one version of baseball uses a ball the size of an orange ("hardball") while another version uses one the size of a grapefruit ("softball"). The rough outline and dimensions of the field on which baseball is played is depicted in Figure A.1. During a playing period, nine players from one team choose to place themselves at strategic points in the field; the X's indicate their typical approximate placement. The player who is located at homeplate faces the 'back' of the field, while the remaining players face the 'front'. There are several "distinguished locations" in the field: firstbase, secondbase, thirdbase, and homeplate. We will say more about these locations soon. Finally, during a playing period, a member of the team which is not in the field stands at homeplate holding a bat while his teammates sit on a bench outside the playing field; this player faces the opposing player at the pitcher's mound.

The ultimate objective of the game is the accumulation of runs. The team with the most runs at the end of the game is declared the winner. A run is defined as the encirclement of the four distinguished bases by a player who initially stands at homeplate holding a bat. In order to 'score

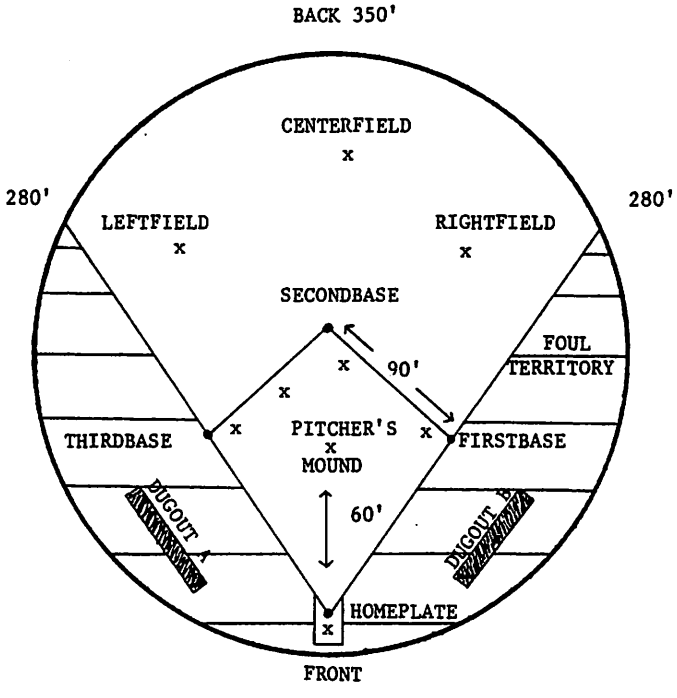


Figure A.1 The Playing Field for Baseball

a run' the bases must be traversed in the following order: firstbase, secondbase, thirdbase, and finally return to homeplate. Obviously, the objective of the opposing team in the field is to prevent batters, in accord with the rules of the game, from the performing this action. Note that a run cannot be scored by a team in the field, but only by the team at bat.

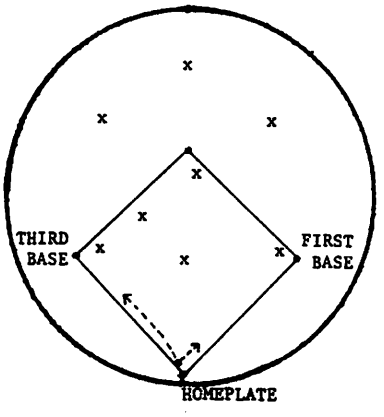
Typically, play proceeds as follows: the player (the pitcher) standing at the pitcher's mound throws the ball toward homeplate and the opposing player (the batter) who is standing there with the bat then attempts to hit the ball. If he is unsuccessful, e.g., he swings the bat and misses the ball or he does not swing at all, then the pitcher's teammate (the catcher), who is crouched behind the batter, catches the ball and throws it back to the pitcher in order for this cycle to repeat. If the batter is successful and hits the ball out into the field in fair territory, then he is allowed to try to run around all the bases. He is not, however, required to circle all the bases in one continuous sequence of running; rather, he may stop at any base, using the base as a sanctuary.

Assume that a batter has hit the ball, and stopped at firstbase. Then, a teammate is allowed to walk from his resting place (the dugout) along the sidelines to homeplate with a bat in order to also have the opportunity to hit the ball; this is a ritual and not governed by any rules in the game. If this second batter hits the ball into fair territory, then his teammate on firstbase can try to advance to other bases while the batter himself attempts to encircle the bases. Note that this second

batter is not allowed to overtake or travel with the previous batter as they circle the bases. Runs are scored as players return to homeplate after circling all the bases.

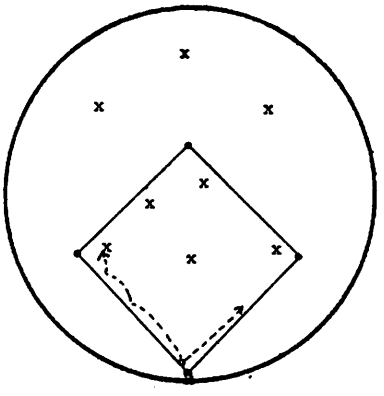
There comes a point when the team at bat relinquishes its opportunity to score; the team in the field returns to their dugout while the team at bat takes up the positions in the field. This switch in roles takes place when the team at bat has made three outs. An out can be executed in a number of different ways. For example, if a player in the field catches the ball after it is hit by the opposing batter and before it touches the ground, then the batter is said "to be out". In effect, an out is a failed opportunity for the batter and his team. Sometimes the team at bat makes three consecutive outs, i.e., three consecutive batters fail to reach firstbase; sometimes many batters -- theoretically, an unlimited number -- can bat before three outs are made. When the team at bat makes three outs, they temporarily turn over the right to bat to the opposing team.

An out may also be made in the following manner. Assume that a batter hits the ball on the ground in the direction of the player at thirdbase (Figure A.2a). The thirdbaseman can catch the ball (Figure A.2b) and throw it to his teammate at firstbase (Figure A.2c), so that his teammate catches the ball before the batter reaches firstbase (Figure A.2d). If the firstbaseman does catch the ball before the batter reaches firstbase, then the batter is said "to be out" and must return to his dugout; he has lost an opportunity to make a score. On the other hand, if



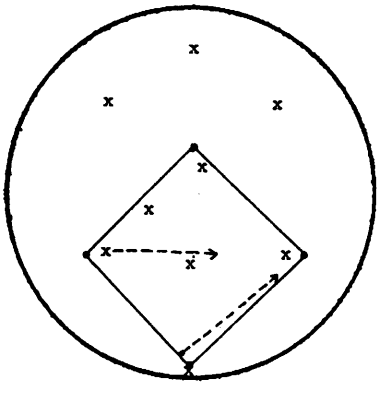
(a)

The batter at HOMEPLATE hits BALL on ground towards THIRDBASE; the batter then runs towards FIRSTBASE (which is a sack lying on the ground).



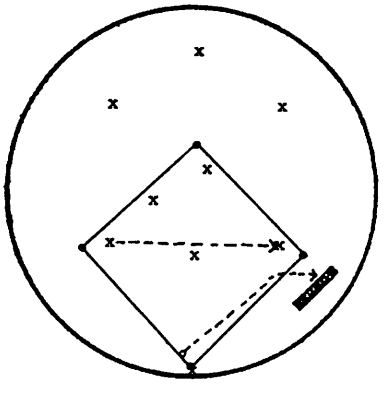
(b)

The player at THIRDBASE catches the BALL, while the batter continues to run towards FIRSTBASE.



(c)

The player at THIRDBASE throws BALL to teammate at FIRSTBASE; batter still running.



(d)

Player at FIRSTBASE catches BALL before batter reaches FIRSTBASE. The batter is not allowed to remain at FIRSTBASE and returns to his resting place in the DUGOUT.

Figure A.2 An Infield Groundout: Another Way to Make an Out

the batter does arrive at firstbase before the opposing player at first-base catches the ball, then the batter is allowed to remain on firstbase; the batter, then still has an opportunity to make a score in subsequent activity. Note that any player in the field can catch the ball and relay it to firstbase.

Clearly, then, one of the batter's strategies is to hit the ball as far as he can, e.g., if the batter hits the ball to the players in the outfield (leftfield, centerfield, rightfield) and the ball is not caught before it touches the ground, then the likelihood is extremely low that the throw from such a location will arrive at firstbase before the batter. Moreover, the batter is not required to stop at firstbase, but may try to advance to second or thirdbase; to be successful, however, he must arrive there before the ball comes into the possession of the opposing player at that base.[52] The strategy of the opposing pitcher is to throw the ball towards the batter in such a way as to prevent the batter from hitting it. For example, a pitcher can throw the ball very fast or can make it do "tricks" such as curve, sink, or flutter. A ball thown in these ways is much more difficult to hit.

Periods of play in baseball are based on outs. In an inning, both the A-team and the B-team each have a turn at bat until they make three outs. There are nine innings in a regulation baseball game. Thus, each team is allowed to bat and score runs nine times during a game. However, if at the end of nine innings the score is tied (both teams have the same score) the teams are permitted to play additional innings , until the score is no longer tied.

There are a number of ways to make outs and a number of other ways besides hitting the ball to advance around the bases. In fact, the rules become quite complicated. However, this thumbnail sketch of baseball should be sufficient to enable a reader to appreciate the system's machinations.

A P P E N D I X B

BRIEF DESCRIPTION OF OUR COMPUTER SYSTEM

It is part of the lore, that one of the rites of passage in AI is the construction of two AI systems -- the first one serving as a learning experience and the second one serving as a finished product. While the personal experience gained in building a large system is invaluable, more needs to be written about these experiences and in this way share some of the insights -- and obstacles -- that arise in this type of endeavor. To this end, we have tried to describe in what follows some of the more distinctive features of our implementation. While we have a few choice words to say about LISP, the spirit of this section is more descriptive than evaluative or theoretical.

As is often the case, memory limitations proved to be the overriding constraint in the design and construction of our system. While early work was done on a CDC 3800 in SNOBOL, the bulk of the system was done on a CDC 6600 in LISP. Since only 51 K was available on the 6600 for user applications, the system needed to be repeatedly segmented. Thus, the overall task and the necessary processing was decomposed into three relatively independent sub-tasks. First, all the snapshots were subjected to Attention Focussing processing. Next, the act-schemas were applied to all the episodes in order to establish the physical relationships between the actions. Finally, the higher level functions were performed on a

third pass through the episodes; parsing, hypothesis generation, class formation (generalization) and prediction matching were all performed on an episode before the system proceeded to the next episode. In this way the system could immediately incorporate newly verified rules into its knowledge base and use them in the analysis of subsequent episodes.

B.1 The Benefits of LISP

The major aspects of our implementation -- Phase 2 and 3 -- were written in a version of LISP [McC65] implemented on a CDC 6600 [KON75]. As is well-known, the data structures, operations, and control structures in LISP are eminently suited for the non-numeric processing required in this type of task. However, it was another feature of LISP which made the most significant contribution to the successful construction of our system -- LISP encourages programming without side-effects. One can write the vast majority of LISP functions without the use of global variables; all a LISP function needs to do is return a value. Desired results can be built up by simple functional composition. Only at the topmost level is it necessary to set the value of some global variable; in our system we used the property-lists of a few global variables (e.g., one for each of the competitive and cooperative hypothesis) to save the results of the computation.

The functions in the system decomposed neatly into a hierarchy of levels. At the highest level there were the bookkeeping functions and

driver functions which generated unique names and stored the results of the computations on the property-lists of those variables; next, there were the predicates which composed the production rules; below that there was a level of functions which could be considered helping functions for the predicates; next, there were the specialized functions which interacted with the associative data base and pattern matcher; and finally, there were the additional list processing functions. Since only the functions at the highest level produced side-effects -- set variables -- functions at one level were defined, written, and quickly debugged independently of any function at a higher level; in fact, it was not uncommon to have whole systems of functions work correctly immediately with no debugging required! In this structure, we were able to quickly redefine functions at one level, since they were primarily composed of functions set at the next lowest level. Moreover, since there were no global variables embedded in these functions we did not worry about unwanted interactions -- they simply could not happen. Of course, this style of programming can be employed with other computer languages. However, we have found that the "assignment statement" is often too seductive, and the strict discipline developed naturally in LISP can be quickly forgotten.

Following our polemic on LISP, we should also mention that our version of LISP was augmented with a PLANNER-like associative data base and pattern matching facilities [HEW72]. These were extensively used and proved invaluable in the conceptualization and implementation of our system.

B.2 Phase 1: Attention Focussing

A FORTRAN program was written which generated snapshots in accordance with the rules of baseball; in a typical game there were approximately 2000 snapshots. These were then filtered and segmented during Attention Focussing. This program was written in SNOBOL. A great deal of pattern matching was required in this phase of processing and SNOBOL is particularly well suited for such a task.

B.3 Phase 2: Application of Act-Schemas

As the appropriate act-schema was applied to an action in an episode, the system generated an internal variable for the action, which served as the repository for the inferences made by the act-schema, e.g., the features PHYSICALLY-ENABLE or PHYSICALLY-ENABLED-BY were added to the appropriate actions. The conditions (PHYSICAL-ENABLING-CONDITIONS, CONSEQUENCES) in the act-schemas were implemented as patterns (see Figure III.4) which were matched on the current episode contained in the associative data base.

B.4 Phase 3

B.4.1 The ATN Parser

The ATN parser, which was used to apply the simple episode grammar, was written in LISP. Its implementation was exceedingly simple and clean, since the LISP stack itself was used to facilitate the depth-first backtracking required by an ATN system. Besides the functions needed to imple-

ment the various edge types (CALL-TO-A-SUBNETWORK, TEST, SET-A-REGISTER), only a handful of auxiliary (driver) functions were needed to construct the ATN parser per se.

While we found that writing ATN grammars was natural, their implementation and debugging was much more difficult. The problems are due to the global nature of the "registers" (variables) in an ATN; changes made to the contents of a register in a subnetwork remain in force when control is returned to the calling network. We have already expounded on the difficulties of side-effect programming, and the virtues of LISP in this regard. Thus, we suggest that a binding scheme analogous to that of LISP's be employed in an ATN setting; this would resolve the global variable problem, yet not destroy the spirit of the ATN formalism.

B.4.2 Causal-Link Schemas

The Causal-Link Schemas were most naturally implemented as production rules. The left-hand side of a CLS consisted of a set of predicates, which if all true, would enable the right-hand side to trigger. The predicates were LISP functions; some were simple and tested for the existence of a feature in the pattern descriptions (e.g., [PHYSICAL-ENABLE ACT1 ACT2]), while others were more complicated and needed to access information in the pattern descriptions, act-schemas, and facts about the spatio-temporal world (e.g., [CAN-EFFECT-PERFORMANCE 'THROW 'SWINGHIT]). In earlier versions of the system, the predicates either returned TRUE, or, instead of simply returning FALSE (NIL), they returned the condition that caused the rule to fail. The hope was that subsequent processing

might be able to make use of this additional information. Unfortunately, there was not sufficient time to pursue this issue, and thus this hope was not realized.

The CLSs stepped through the actions in an episode under the control of the ATN parser. The actual application of the CLSs as production rules was rather trivial; inasmuch as our LISP interpreter could interpret the IF-THEN construction, it served as the production rule interpreter. Unlike more traditional production rule systems, ours did not have a data base for the storage of intermediate results. Rather, when a CLS was triggered the system generated an internal variable name for the hypothesis; the property list of this variable served to hold the pattern description representation, the production rule representation, the predictions based on the hypothesis, the confidence value of the hypothesis, and miscellaneous bookkeeping information.

B.4.3 Generalization

Due to memory limitations, once the ATN parser and CLSs had processed an episode, they were overlayed by the generalization routines. In this stage of processing, the current episode was matched against episode classes which had already been formed. If a class was found to be similar to the episode under examination, (i.e., both episodes had the same set of competitive and cooperative CLSs) then the pattern descriptions in a class were generalized to accomodate the differences in the new episode. Since the hypotheses were represented both as pattern descriptions and production

rules, there was some simple bookkeeping which needed to be performed. Namely, the appropriate production rule and predictions were generalized to correspond to the pattern descriptions. If an episode had not appeared before, another name was generated internally, and its property-list was used to store pointer information. Similar processing was performed to produce the more abstract classes based only on the final competitive goals in the episode.

B.4.4 Predictions and Evaluation

Once the generalization phase was complete, the predictions and associated evaluation procedures were brought into memory and overlayed the generalization procedures. Each prediction, implemented as a production rule, was then applied to the episode in the associative data base; matches resulted in the modification of the confidence value of the appropriate hypothesis.

The confidence values were monitored. If a Type-II prediction were matched, then the corresponding hypothesis was eliminated. On the other hand, if the confidence value of the hypothesis reached threshold, then it became a truth. Such acquired CLSs were added to the pool of CLSs and were used in the analysis of subsequent episodes. Also, when a hypothesis was accepted as true, the system would check the unverified hypotheses and modify the confidence values of those which were inconsistent with the new truth and those which were consistent with it.