Massachusetts Institute of Technology
Artificial Intelligence Laboratory
AI Memo 405

NUDGE, A KNOWLEDGE-BASED SCHEDULING PROGRAM[1]

Ira P. Goldstein and R. Bruce Roberts

February 1977

Traditional scheduling algorithms (using the techniques of PERT charts, decision analysis or operations research) require well-defined, quantitative, complete sets of constraints. They are insufficient for scheduling situations where the problem description is ill-defined, involving incomplete, possibly inconsistent and generally qualitative constraints. The NUDGE program uses an extensive knowledge base to debug scheduling requests by supplying typical values for qualitative constraints, supplying missing details and resolving minor inconsistencies. The result is that an informal request is converted to a complete description suitable for a traditional scheduler.

To implement the NUDGE program, a knowledge representation language -- FRL-0 -- based on a few powerful generalizations of the traditional property list representation has been developed. The NUDGE knowledge base defined in FRL-0 consists of a hierarchical set of concepts that provide generic descriptions of the typical activities, agents, plans and purposes of the domain to be scheduled. Currently, this domain is the management and coordination of personnel engaged in a group project.

NUDGE constitutes an experiment in knowledge-based, rather than power-based AI programs. It also provides an example of an intelligent support system, in which an AI program serves as an aid to a decision maker. Finally, NUDGE has served as an experimental vehicle for testing advanced representation techniques.

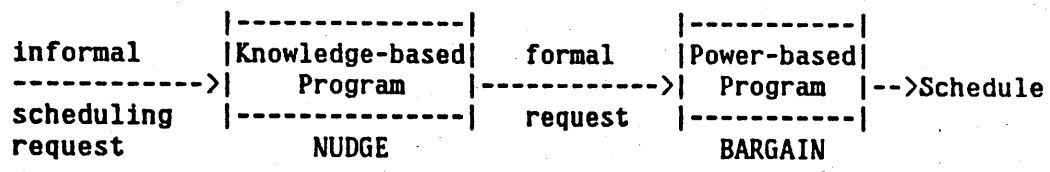
This report describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. It was supported in part by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contract N00014-75-C-0643.

[1] This paper has been submitted to the Fifth International Conference on Artificial Intelligence.

1. INTRODUCTION

A classic issue in AI is the knowledge versus power controversy [Minsky & Papert 74]. The knowledge position advocates that intelligence arises mainly from the use of a large store of specific knowledge, while the power theory argues for a small collection of general reasoning mechanisms. This paper reports on an experiment in which a knowledge-based program NUDGE has been implemented for the scheduling domain, a domain in which power-based programs have long been the dominant paradigm.

Traditionally, scheduling programs apply simple but powerful decision analysis techniques to finding the optimal schedule under a well-defined set of constraints. The performance of NUDGE confirms that for well-defined, formal situations, the traditional power-based approach is appropriate. But for the problem of defining these formal situations when given only informal specifications, a knowledge-based approach is necessary. By an informal specification, we mean a scheduling request that is potentially incomplete, possibly inconsistent and qualitative. (See Balzer [74] for an analysis of informal program specifications.) Thus, the NUDGE program accepts informal requests and produces a calendar containing possible conflicts and an associated set of strategies for resolving those conflicts. A domain-independent search algorithm BARGAIN then resolves these conflicts by traditional decision analysis techniques.



NUDGE uses a broad data base of knowledge to expand and debug informal scheduling requests. The database is used to supply missing details, resolve inconsistencies, determine available options, notice necessary prerequisites and plan for expected outcomes. To manage this large store of knowledge, a representation language -- FRL-0 -- has been implemented. FRL-0 extends the traditional attribute/value description of properties by allowing properties to be described by: comments, abstractions, defaults, constraints, indirect pointers from other properties, and attached procedures.

These are not new representation techniques. Abstraction, for example, was

discussed in Quillian [68], and attached procedures have become a common property of AI languages since PLANNER [Hewitt 69]. However, the strengths and weaknesses of these representation techniques and their potential interactions is still not well understood. For this reason, we have chosen not to include as many representation capabilities as are currently being implemented in KRL [Bobrow and Winograd 76] and OWL [Martin 77]. We view FRL-0 as an experimental medium to study the utility of a few specific capabilities and their interactions.

Because a knowledge-based approach requires a large store of specific data, it was necessary to choose a particular domain to carry out our experiment. Our criterion was to select a realm in which scheduling requests are typically informal. This criterion ruled out such scheduling problems as those of an assembly line. (See Tonge [63] for an AI treatment of this problem.) Instead, we selected office scheduling; in particular, assisting a manager in scheduling his team. This environment includes scheduling meetings, monitoring the progress of subgoals assigned to team members, alerting the manager to deadlines, and real-time rescheduling.

In providing NUDGE with the knowledge necessary for these functions, our research serves a third purpose beyond (1) exploring the relation between knowledge-based and power-based scheduling and (2) exercising various representation strategies. It provides insight into the categories of knowledge that are necessary for office scheduling (independent of their representation). NUDGE contains a hierarchy for activities involving information transfer, for people in various roles related to this transfer, for the plans governing these transfers, and for the associated demands on time, space and personnel. The hierarchy is on the average five levels deep and includes approximately 100 objects, each described by a generalized property list called a frame. An abridged version of this hierarchy appears below, with specialization indicated by nesting.

```
(THING (ACTIVITY (EATING (LUNCH) (DINNER))
                 (TRANSPORT (FLY) (DRIVE) ...)
                 (MEETING (PA-MEETING)
                          (COMMUNICATION-AT-A-DISTANCE (PHONECALL)
                                                       (CORRESPONDENCE (US-MAIL)
                                                                       (NET-MAIL)))
                          (ONE-WAY-COMMUNICATION (READ)
                                                 (WRITE (DRAFT) (EDIT))
                                                 (TELL (LECTURE))
                                                 (LISTEN)))
                 (RESEARCH (PA-PROJECT) (VLDB-PROJECT) ...))
       (PEOPLE (GROUP (AI-LAB (VLDB-GROUP) (PA-GROUP) ...) ...)
               (PERSON (PROFESSOR (IRA))
                       (STUDENT (GRADUATE-STUDENT (MITCH) (CANDY))
                                (UNDERGRADUATE-STUDENT))
                       (VISITOR (ACADEMIC-VISITOR)
                                (GOVERNMENT-VISITOR)
                                (REPORTER))))
       (PLACE (CITY (CAMBRIDGE) (SAN-FRANCISCO) ...)
              (SCHOOL (MIT) (STANFORD) ...)
              (BUILDING (HOUSE (25-WILDWOOD))
                        (OFFICE-BUILDING (545-TECHNOLOGY-SQUARE))
                        (RESTAURANT (DODIN-BOUFFANT)))
              (ROOM (OFFICE (NE43-819))
                    (SEMINAR-ROOM (AI-LOUNGE))))
       (TIME (INTERVAL) (MOMENT))
       (OBJECT (INFORMATION (REPORT (PROPOSAL)
                                    (PROGRESS-REPORT)
                                    (FINAL-REPORT))
                            (LETTER)
                            (MEMO))
               (VEHICLE (CAR) (PLANE) ...)
               (REMINDER)
               (CALENDAR)))
```

The term "frame" as used in FRL-0 was inspired by Minsky's [75] development of frame theory. Frame theory contends that (1) intelligence arises from the application of large amounts of highly specific knowledge, as opposed to a few general inferencing mechanisms, and (2) this is accomplished through the use of a library of frames, packets of knowledge that provide descriptions of typical objects and events. These descriptions contain both an abstract template providing a skeleton for describing any instance and a set of defaults for typical members of the class. The defaults allow the information system to supply missing detail, maintain expectations, and notice anomalies.

We have yet to investigate with equal care related areas of knowledge not strictly involved in scheduling the information flow between members of a research team -- this includes space allocation, budgeting, and travel scheduling. The last of these is the

focus of the GUS system developed by Bobrow et al [76]. GUS is a frame-based dialog system that addresses many of the issues raised here. The major difference is that GUS focusses on the use of frame representations to support dialog comprehension while NUDGE focusses on their use to support sophisticated scheduling.

Finally, although developed for the office scheduling domain, the NUDGE knowledge base does have a broader applicability. The use of a hierarchy of successively more abstract concepts allows highly specific information to be clearly segregated from more general facts about the world. Furthermore, some of the concept hierarchies such as those for time and place apply *in toto* to many other applications.

2. SCENARIO

The following request is typical of those understood by NUDGE.

(S1) > NUDGE, schedule a meeting with Bruce for next Tuesday.

S1 is an informal specification in that such facts as the following are left unsaid:

Where the meeting is to be held.
Which Tuesday is next Tuesday.
What time on Tuesday is preferred.
How long the meeting is should last.
Whether there are any prerequisites.
What to do if Bruce is unavailable.

As such it is typical of the requests made by people of one another. Given a knowledge of the individuals involved, their roles and the context, NUDGE expands S1 to the more complete S1', where the added information is in italics.

(S1') > I UNDERSTAND YOUR REQUEST TO MEAN: SCHEDULE A MEETING WITH BRUCE *AND ME AT 3 IN THE AFTERNOON* NEXT TUESDAY *ONE WEEK FROM TOMORROW LASTING ONE HOUR IN MY OFFICE TO DISCUSS THE PERSONAL ASSISTANT PROJECT*.

When processed by the the language comprehension component of NUDGE [Bullwinkle 77], S1 causes the construction of a frame gestalt, a data structure of instantiated frames containing all information in S1'. (The actual text of S1' is generated by the text output procedures associated with the frames in this gestalt.)

The knowledge-based phase of the scheduling concludes with the conversion of the information in the frame gestalt to a calendar containing possible conflicts and an associated set of strategies for resolving those conflicts. The BARGAIN program then reports the highest utility compromise to the user.

(R1) YOUR PREFERRED TIME FOR MEETING WITH BRUCE ON TUESDAY IS ALREADY FILLED BY A MEETING WITH MARK. HOWEVER, MARK'S PURPOSE IS TO DISCUSS LANGUAGE RESEARCH AND CANDY IS AVAILABLE. SHALL I RESCHEDULE MARK TO SEE HER INSTEAD.

Mark's purpose is not specifically to see Ira, and hence, a member of Ira's group can be substituted to see Mark. Observe that sufficient knowledge has made it unnecessary to juggle the possible time of the meeting between Ira and Bruce, producing a compromise not entirely satisfactory to either. Instead, a sufficiently broad perspective has allowed NUDGE to identify the option of finding an alternative participant by referencing the purpose of the conflicting appointment. A knowledge-based phase in the scheduling thus avoids the myopia a purely power-based approach can exhibit.

Continuing the scenario:

(S2) > Yes.

(R2) THE MEETING WITH BRUCE IS SCHEDULED. SHOULD BRUCE PREPARE A WRITTEN PA PROGRESS REPORT?

(S3) > Yes.

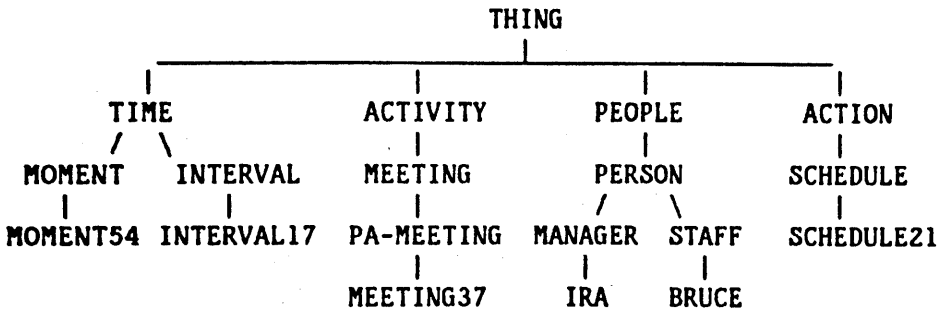
(R3) I WILL REMIND BRUCE ABOUT THE REPORT.

Section 3 describes the frame gestalt representing NUDGE's completion of the S1 request. Section 4 discusses the representation techniques of FRL-0 supporting the creation of the gestalt. Section 5 analyzes the underlying knowledge of activities and individuals represented with these techniques.

3. THE FORMATION OF FRAME GESTALTS

A frame gestalt consists of a set of generic frames, instantiated appropriately for a particular scheduling request. The generic frames are selected on the basis of clues implicit in the scheduling request; knowledge-based reasoning begins with a recognition process. Information missing in the request is then computed from defaults, constraints, and procedures -- all associated with these generic frames. For S1, the gestalt includes the frames shown below, with daughters of a frame representing its specializations. Many other interrelations among these frames are not shown.

```
                                THING
          ____________________________|____________________
          |                 |              |              |
        TIME             ACTIVITY       PEOPLE         ACTION
        / \                 |              |              |
  MOMENT   INTERVAL     MEETING        PERSON        SCHEDULE
    |        |            |            /    \           |
MOMENT54 INTERVAL17  PA-MEETING  MANAGER  STAFF    SCHEDULE21
                          |          |       |
                      MEETING37     IRA    BRUCE
```



The goal of the knowledge-based phase is to compute this frame gestalt.

The input to the gestalt formation process is a set of partially instantiated frames representing the information actually present in the informal request. This input is generated by a natural language front end consisting of the Wait-and-See-Parser developed by M. Marcus [76] and a frame-based semantics designed by C. Bullwinkle [77]. There are four partially instantiated frames: MEETING37, SCHEDULE21, INTERVAL17, MOMENT54.

MEETING37 is the frame for the proposed meeting and initially contains information regarding the participants and time of the event extracted from the English request. We give MEETING37 below in FRL-0 notation.

```
(MEETING37
  (AKO    ($VALUE ( PA-MEETING    (SOURCE: PRAGMATICS S1)))
  (WHO    ($VALUE ( IRA           (SOURCE: PRAGMATICS S1))
                  ( BRUCE         (SOURCE: SEMANTICS S1) )))
  ...)
```

MEETING37 reports that the participants are Bruce and Ira. Bruce is known from the semantic interpretation of S1, while Ira is inserted on the basis of pragmatics, i.e. that the person requesting the meeting wishes to be included among the participants. MEETING37 has been identified by pragmatics as A-KIND-OF (AKO) PA-MEETING on the basis of knowledge regarding the common activities of Ira and Bruce. Had no such special knowledge existed, the sentence would simply have triggered an instance of the MEETING frame.

The first element of these frame structures is the name of the frame name and each remaining item is a <u>slot</u>. Each slot has a value, explicitly marked by $VALUE. A slot can have more than one value, as in the participant slot of MEETING37. A frame is thus simply a multi-level association list. The semantics of attached procedures and

inheritance is enforced by the access functions.

4. REPRESENTATION TECHNOLOGY

The formation of a frame gestalt occurs by expanding the the frames extracted from the initial request in terms of the knowledge stored in the FRL database. This section discusses this process in terms of the contributions made by six representations techniques embedded in FRL-0: comments, abstraction, defaults, constraints ind ection and procedural attachment.

(1) Comments. The first generalization of property lists in FRL-0 is the inclusion of comments attached to values. Comments are used in these examples to record the source of the value in each slot. So far, the only *source* is the semantic and pragmatic interpretation performed by the language comprehension process. Alternative sources are inferences made by attached procedures and inherited properties. Other kinds of commentary provide numerical utilities for use by the scheduler, and describe differences between multiple procedural methods attached to the same slot.

This commentary provides guidelines both for BARGAIN to judge the reliability and and strength of various constraints and for NUDGE to debug inconsistencies arising from conflicting contributions by different frames during the gestalt formation process. The former would arise if conflicts existed between the generic information of the IRA and BRUCE frames, while the latter is exemplified by S1 being part of a dialog. The sentence "The meeting should be tomorrow in my office." would override the default suggested by generic knowledge. Our use of commentary to guide a debugging process derives from research by Sussman [73] and Goldstein [74].

Self-knowledge, in the form of machine understandable annotations, also faciliates the systems ability to explain its inferences to a user. This is critical if the user is to become confident in the system's capabilities.

(2) Abstraction. The second property list generalization is to allow infor ıtion to be inherited between concepts. In essence, this is an implementation .f Quillian's SUPERCONCEPT pointer, but between property lists rather than nodes in a sem. ic net. For example, details of the meeting between Bruce and Ira are inherited from generic

description of PA-MEETINGs. This description includes answers to such questions as where such meetings are typically held, when, why, who is involved, what prerequisites are required and what consequences result.

Since the answers to some of these questions is clearly applicable to a broader set of activities than meetings of the PA-GROUP, the information is distributed in a hierarchy of successively more general frames, thereby achieving both power and economy.

Each frame points to its generalization by means of its AKO slot. The process by which information in a generic frame is acquired by a specialized instance of that frame is called inheritance. MEETING37, for example, inherits information from its generalization, the PA-MEETING frame.

```
(PA-MEETING
  (AKO    ($VALUE   ( MEETING )))
  (WHY    ($VALUE   ( PA-PROJECT )))
  (WHERE  ($DEFAULT ( AI-PLAYROOM )))
  (WHEN   ($DEFAULT ( ((ON FRIDAY) (FOR 1 HOUR)) )))
  (WHO    ($DEFAULT ( (IRA   (ROLE: MANAGER))
                      (BRUCE (ROLE: FRL))
                      (CANDY (ROLE: SEMANTICS))
                      (MITCH (ROLE: SYNTAX)))))
  ...)
```

(3) Defaults. The third generalization that naturally accompanies a hierarchy of frames is default information, as the slots of a generic activity typically supply default answers to the common questions asked about such events. The utility of such default information was a major insight of Minsky's original frames research. Their use is prevalent throughout the NUDGE database, and give the scheduler much of its power.

For example, PA-MEETING supplies the information that such activities typically involve four people, occur on Fridays in the AI Lab Playroom and last one hour. The $DEFAULT atom distinguishes defaults from values. We shall refer to the different kinds of information associated with a slot as its facets. The role commentary information associated with the participants of PA-MEETINGs is used by the PERSON-SWAPPING scheduling strategy of the BARGAIN program. (Mangers are optimistically defined to know all of the team members' roles.)

In forming a frame gestalt, the defaults of superior frames are used unless they are overridden by information from a more reliable source, such as the explicit constraints

of the original request. Thus, the WHERE default would apply to the MEETING37 gestalt. The WHEN default, however, is overridden by the explicit request in S1 that the meeting occur on Tuesday.

Defaults are also useful to the natural language understanding system by supplying expectations that aid the parser and semantics in processing ambiguity and ellipsis. However, we do not develop that application here.

(4) Constraints. A knowledge representation language must accommodate descriptions of properties if it is to support the recognition of new instances of a generic concept. FRL-0 allows constraints to be attached to a slot by means of facets for requirements and preferences. These constraints are illustrated in the MEETING frame, which is the immediate generalization of PA-MEETING.

```
(MEETING
  (AKO   ($VALUE   ( ACTIVITY )))
  (WHO   ($REQUIRE ( (EXISTS ?WHO (HAS-ROLE ?WHO 'CHAIRMAN)) )))
  (WHEN  ($PREFER  ( (< (DURATION ?WHEN) (HOUR 1.5)) )))
  ...)
```

Requirements are predicates which must be true of the values in a slot. Preferences can be relaxed yet leave a valid slot. The $REQUIRE facet of MEETING stipulates that a chairman be present at all meetings. The $PREFER facet states that meetings should not last longer than 90 minutes.

It is possible to collapse defaults, preferences and requirements into a single CONSTRAINT facet. However, we have found it convenient to preserve the distinction, given the use of these facets by a scheduler. Requirements cannot be relaxed. Preferences, however, can be negotiated. Defaults are preferences that offer specific alternatives, rather than acting as predicates on a set.

KRL's development of "perspectives" in which a frame is described from a particular viewpoint is a more elaborate kind of constraint than we typically employ. While a frame pattern matcher can, in essence, specify a perspective, we have not generally used such a device for the narrowly defined world of office scheduling. Whether an extended description mechanism will be needed for more complex applications remains to be seen. Our current plans are to see where FRL-0 fails before we incorporate more powerful, but more complex techniques.

(5) Indirection. Not all of the relevant description of MEETING37 is contained in abstractions of the meeting concept. Frames in orthogonal hierarchies also supply constraints. For example, activities involve agents, and the frames for these agents have relevant information. The frame system has separate hierarchies for activities and people, interconnected by indirection. The IRA frame exemplifies this.

```
(IRA
  (AKO              ($VALUE ( PERSON )))
  ((MEETING WHEN)    ($PREFER  ( (DURING AFTERNOON) )
                               ( (ON FRIDAY) )))
  ((MEETING WHERE)   ($DEFAULT ( NE43-819 ))
                     ($PREFER  ( (IN 545-TECH-SQUARE) )))
  ((PA-MEETING WHEN) ($DEFAULT ( (AT 3 PM) )
                               ( (AT 10 AM) ))
                     ($PREFER  ( (ON TUESDAY) )))
  ...)
```

The first atomic slot identifies Ira as a person. The remaining non-atomic slots (ie, slots with compound names) provide information regarding various activities with which IRA is typically involved. For example, in general IRA prefers MEETINGs on Friday afternoons in his office. This information is not stored directly in the activity frame since it not generically true for the activity, but only applicable when IRA is involved. IRA's default of NE43-819 as the place to hold meetings supplies the value of the WHERE slot of MEETING37.

The indirect information appears in the frame gestalt if both the agent and activity frames are triggered. For S1, triggering IRA and MEETING together resulted in the frame gestalt supplying the missing information regarding the location of the meeting. Thus, indirection provides links between different hierarchies, extending the frame system to include a network of contingent facts.

Indirection is a simplified kind of mapping between concepts. FRL differs from MERLIN [Moore & Newell 73] (in which general mapping is allowed) by providing a restricted but more structured environment. Mapping occurs, in essence, only between agent and activity frames through indirection and between concept and superconcept frames through inheritance.

(6) Procedural attachment. A knowledge representation must allow procedural as well as declarative knowledge. Procedural attachment provides this capability in FRL-0. This capability is found in most AI languages beginning with PLANNER [69].

There are typically three kinds of procedural attachment, and all are provided in FRL-0. These are if-added, if-needed, and if-removed methods. A difference from traditional AI languages is that these procedures are attached to the slots of a frame rather than to assertions of an arbitrary form. FRL-0 is thus a more structured environment than languages like PLANNER and CONNIVER [McDermott & Sussman 72]. Providing a mechanism for triggering arbitrary procedures by adding a value to a slot supports the fundamental operation of FRL-0 which is instantiation; that is, creating an instance of a frame and filling in values.

For example, when the time of MEETING37 is arranged (a value of the WHEN slot is assigned) its name is entered in a calendar for easy reference. The method for doing this is supplied by the WHEN slot of the ACTIVITY frame.

```
(ACTIVITY
  (AKO          ($VALUE      ( THING )))
  (WHO          ($REQUIRE    ( (ALL (AKO PERSON)) ))
                ($IF-NEEDED  ( (ASK) (TYPE: REQUEST) )
                             ( (USE TOPIC) (TYPE: DEDUCE) )))
  (WHEN         ($IF-ADDED   ( (ADD-TO-CALENDAR) ))
                ($REQUIRE    ( (AKO INTERVAL) )))
  ...)
```

ACTIVITY also illustrates IF-NEEDED methods. These methods allow access to arbitrary procedures for supplying values. For example, examine the WHO slot of ACTIVITY. There are two IF-NEEDED methods there. The first, (ASK), is a function that requests the value from the user. Its purpose is indicated by the comment TYPE: REQUEST. The other method, (USE TOPIC), attempts to deduce the participants by accessing the WHO slot of the frame provided as the value of the TOPIC. The comment on this method indicates that it is of TYPE: DEDUCE. The TYPE comments are used by the function controlling the overall instantiation process (the IF-NEEDED method of INSTANCE in THING, which all frames inherit). Their function is to allow deductive methods to be used in preference to interactive requests if possible.

If-needed methods are exceedingly powerful. For example, defaults can be viewed as

a special kind of if-needed method, so useful and widespread that a special facet of a slot is devoted to it. Idiosyncratic forms of inheritance (using other than the AKO link) can be imbedded in an if-needed for appropriate slots.

Attached procedures are also used to maintain the integrity of the database. For example, AKO and INSTANCE are slots that provide a two-way link between a frame and its generalization. This linkage is maintained by a pair of if-added and if-removed methods. The procedures which implement this mechanism appear in THING, the most general frame in NUDGE.

```
(THING
  (AKO          ($IF-ADDED    ( (ADD-INSTANCE) ))
                ($IF-REMOVED  ( (REMOVE-INSTANCE) )))
  (INSTANCE     ($IF-NEEDED   ( (INSTANTIATE-A-FRAME) ))
                ($IF-ADDED    ( (ADD-AKO) ))
                ($IF-REMOVED  ( (REMOVE-AKO) )))
  ...)
```

Subtleties. We conclude our discussion of FRL-0 as a representation language with a consideration of first some of the subtleties involved in the use of these six techniques and then some of FRL-0's current limitations.

1. There is more than one kind of inheritance.

Additive and restrictive inheritance are two kinds of inheritance strategies that correspond to two common forms of specialization. Additive inheritance is appropriate where specialization adds new non-contradictory facts to the more general concept. Restrictive inheritance is appropriate where specialization overrides the information contained in the more general concept. Commentary is employed to inform the inheritance mechanism whether to stop or to continue up an AKO chain once the first datum is found.

2. Methods can conflict.

Procedural attachment can be complex. For example, care must be taken to avoid loops: a method in slot A may add a value to slot B that in turn adds a value to slot A. An endless loop results. We handle this by using comments to describe the source of information. It is up to the individual method to access this commentary.

Scheduling multiple methods associated with a single slot may be required, when the methods have an a implicit order. Currently, the frame system executes the methods in a

fixed order. If more subtle ordering is required, the user must combine the methods into a single procedure, with this method responsible for performing the proper ordering.

3. The distinction between value and requirement is not sharp.

Requirements have been presented as predicates to filter out unwanted values. To the extent that NUDGE can reason directly from them, however, they can be used in place of values. For example, an IF-NEEDED procedure can use the requirement to select the generic category when instantiating a new frame to fill a slot.

4. A frame is more than the sum of its parts.

In our initial conception of a frame system, we did not provide for a SELF slot to contain idiosyncratic information about the frame itself. Our hypothesis was that all of the information in the frame could be represented in the individual slots. However, the need arose to represent global information about the frame, not local to any slot. Two examples are knowledge of how to print the frame in English and knowledge of the preferred order in which to instantiate the slots of the frame. For these reasons, a SELF slot was introduced with a set of facets appropriate to the various classes of global information it contains. At present these area a $DISCUSS facet which contains a procedure for describing the frame in prose, and an $ORDER facet which contains a procedure that orders the slots at the time of instantiation.

5. Inheritance of values and defaults can conflict.

A given slot may have both a default and a value in its generalization frame. Which should dominate? Currently, the frame system treats values as more important than defaults, so all of the values of superior frames are checked before a default is accepted. However, this may not be appropriate in all cases. When it is not, the user of the frame system can obtain complete control by asking for the full heritage of the slot, i.e. all of the facets for the slot in the current frame and its superiors. The user can then select the desired datum.

Limitations. The following are limitations of version 0 of FRL.

1. No provision is made for multiple worlds in the frame system, although the BARGAIN program can consider alternative calendars.
2. Procedures cannot be attached to arbitrary forms, but only to values. For example, there is no way to have a procedure trigger when a new requirement is added.
3. Arbitrary data structures cannot be asserted. Only information of the form "frame, slot, facet, datum, comment" can be placed in a frame.
4. Hash coding is not currently used. Hence, it is expensive to find all the frames with a slot of a given name and even more expensive to find all the frames in which a given value appears.
5. Comments cannot be associated with arbitrary parts of a frame, but only either with individual data or the SELF slot of the frame. There is no way to associate a comment with a subset of the slots.
6. Mapping between frames is restricted to matching slots. A generalized mapping function, as in MERLIN [73] wherein one slot can be mapped to another, is not allowed.

Eventually, we may find that the more sophisticated capabilities of CONNIVER, MERLIN, KRL, or OWL are needed. But the rapid rise and fall of PLANNER argues for caution in the introduction of complex new techniques. We plan to introduce additional techniques only as the simple generalized property list scheme of FRL-0 proves inadequate. At present, FRL is adequate to represent the knowledge described in the next section which comprises the basis of its scheduling expertise.

5. EPISTEMOLOGY OF SCHEDULING

NUDGE's ability to expand informal scheduling requests arises from the recognition of the request as an instance of various generic frames. This section provides snapshots of this generic knowledge, which includes frame hierarchies for activities, people, time, and plans.

(1) Activities. Most activities known to NUDGE involve information transfer. They span a set of events central to the scheduling domain and interesting in the subtleties they introduce for successfully debugging conflicting schedules. The "activity" subtree of the frame hierarchy shown earlier illustrates these information transfer activities and their disposition along generalization chains.

The use of concept hierarchies is an alternative to the unstructured set of primitives proposed by Schank [73]. We find this approach powerful in that it facilitates the recognition of new objects and allows fine distinctions to be readily made. To illustrate the first point, S1 could have been treated as referring to an instantiation of the MEETING frame, in the absence of recognizing a meeting between Bruce and Ira as a PA-MEETING. Later, if additional information allows this recognition, the AKO pointer of MEETING37 would simply be adjusted to point to PA-MEETING. Otherwise no change is needed.

The fine distinctions between information transfer activities represented in the activity hierarchy guides the time of scheduling, the preparations required, and the pattern of the consequences. A phone call need not be scheduled for a precise time while a formal meeting must. We find a separate frame useful, rather than representing phone and mail as different instruments to a single communication activity (as might be Schank's approach) because other information is clustered around the choice of means. A letter requires more preparation time than a phone call, implies a certain delay in communication, and leaves open whether a response will be received.

(2) People. Beyond the straightforward record of pertinent facts about a person -- their name, address, phone number, office -- lies the need to capture the alternate roles people play in different situations. Roles exists in their own abstraction tree. For scheduling purposes, the roles define an order of importance that dictates, in the case of conflicting defaults and preferences, the order in which they should be relaxed. Acquiring properties by virtue of playing a role is identical to inheriting information as a specialized instance of a frame; the AKO/INSTANCE links define the path along which this information flows. A particular feature of roles is that they are often transitory and conditional on the type of activity.

Originally, we maintained a strict hierarchy in FRL with each frame pointing to a single abstraction. Recently, we have allowed multiple AKO links. The motivation was that it appeared natural for a person to inherit from several roles. For example, in a given situation, Ira may be both a visitor and a professor. Insofar as the information inherited from multiple parents is non-conflicting, no difficulty arises from this bifurcation in the AKO path. If there is a conflict, it shows up in the frame gestalt with cource comments indicating the various sources of the conflicting information. It is up to the process using the gestalt to decide on a resolution.

For example, as a professor, Ira's preferences with respect to a meeting time would take precedence over a student. As a visitor to another university, they would not. The techniques for debugging schedules take account of this, treating the VISITOR role as overriding the PROFESSOR role.

People can be members of groups. A group has many of the same characteristics as a person insofar as it can appear in the WHO slot of an activity and may have its own "personal" facts: a name, an address, etc. The MEMBER and AFFILIATE slots record this dual relation in NUDGE.

(3) Time. In our earliest work on NUDGE, time was represented simply as points on a real-number line. This was adequate for the formal analysis made by the scheduler, but proved insufficient to represent the informal time specifications supplied by users. People's time specifications are generally incomplete and occasionally inconsistent. To handle these informal requests, we moved to a frame-based representation for time similar to the one described by Winograd [75].

Below is part of the generic frame for a moment in time:

```
(MOMENT
  (AKO          ($VALUE      ( TIME )))
  (MINUTE       ...)
  (HOUR         ($IF-NEEDED ( (ASK) (TYPE: REQUEST) ))
                ($IF-ADDED  ( (DAYTIME-EQUATION) ))
                ($REQUIRE   ( (INTEGER-RANGE 0 23) (TYPE: SYNTAX) )
                            ( (DAYTIME-AGREEMENT) )))
  (DAY          ($IF-NEEDED ( (ASK) (TYPE: REQUEST) ))
                ($IF-ADDED  ( (WEEKDAY-EQUATION) ))
                ($REQUIRE   ( (INTEGER-RANGE 1 31) (TYPE: SYNTAX) )
                            ( (DAY-MONTH-AGREEMENT) )
                            ( (WEEKDAY-AGREEMENT) ))
                ($DEFAULT   ( (CALENDAR-DAY (NOW)) )))
```

```
(WEEKDAY    ($IF-NEEDED ( (ASK) (TYPE: REQUEST))
                        ( (WEEKDAY-EQUATION) (TYPE: DEDUCED) ))
            ($REQUIRE   ( (WEEKDAY?) (TYPE: SYNTAX) )
                        ( (WEEKDAY-AGREEMENT) (TYPE: DEDUCED) )))
(DAYTIME    ($IF-NEEDED ( (ASK) (TYPE: REQUEST) )
                        ( (DAYTIME-EQUATION) (TYPE: DEDUCED) ))
            ($REQUIRE   ( (DAYTIME?) (TYPE: SYNTAX) )
                        ( (DAYTIME-AGREEMENT) (TYPE: DEDUCED) )))
(MONTH              "similar to day"          )
(YEAR           "similar to day and month"    ))
```

The MOMENT frame can represent an incomplete request by creating an instance with only a few of the slots instantiated. The attached procedures -- WEEKDAY-EQUATION and DAYTIME-EQUATION -- derive as many additional descriptors as possible.

A set of time predicates (BEFORE, DURING, AFTER, etc.) have been implemented that allow a user to ask questions regarding his calendar. For example, using the natural language front end, the system can be asked: "Is Ira free on Monday, February 7?"

The predicates take instantiated time frames as input and perform their analysis on a "need to know" basis. That is, (BEFORE M1 M2) will return T even if M1 and M2 are incomplete, providing there is sufficient information to make the required judgment. For example, M1 may specify a moment in January without saying precisely when and M2 a moment in February. In this case, the BEFORE question can be answered despite ignorance of the exact time involved. In this fashion, we go beyond Winograd's original discussion of frame-based time-representations in that he did not consider the issues raised by reasoning with incomplete time specifications. Of course, the nature of the incompleteness may be such that no answer is possible. In this case, the time predicates report that the frames are too incomplete for an answer.

The time predicates can also tolerate a certain degree of inconsistency. For example, suppose a user asks if a meeting is possible with Bruce on Tuesday, January 20, 1977. In fact, January 20 is Monday. But if the frame system knows that Bruce is on vacation all of January, it is more appropriate for it to reply: "I assume you mean Monday, January 20. Bruce will be on vacation then." rather than first asking for a clarification and then telling the user his request will fail.

Inconsistency is detected by IF-ADDED methods which, in the course of deriving values, observe that they have computed a slot value that conflicts with a user supplied

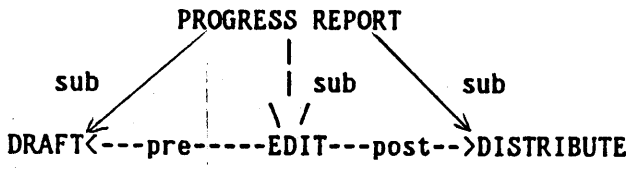
value. A comment regarding the inconsistency is placed both at the slot level and at the frame level. For example, the MOMENT frame for the inconsistent time specification given above would be:

```
(MOMENT12
    (WEEKDAY ($VALUE ( TUESDAY (SOURCE: USER) )
                     ( MONDAY  (SOURCE: DERIVED) )))
    (SELF    (LOGICAL-STATE ($VALUE ( INCONSISTENT (SEE: WORKDAY) )))))
```

The time predicates report the inconsistency, and then attempt to answer the original question by reducing the inconsistency to an incompleteness. This is done by referencing an ordering on the slots corresponding to their relative reliability. Year dominates Month which dominates Day which dominates Weekday. The inferior slot values are ignored until the inconsistency is removed. The question is then answered using the resulting incomplete frame. At best, the time predicates have guessed correctly and the user has learned the answer to his question. At worst, he is alerted to the consistency and responds with a repetition of his original request with the inconsistency removed.

(4) Plans. It is uncommon to schedule isolated events. Typically, clusters of related activities are organized around a theme; a series of meetings to discuss the state of a group's research, work by several people on a joint paper. These clusters embody two kinds of interrelations in addition to the AKO/INSTANCE bond already discussed. First, there is a logical ordering of activities, which in the realms of scheduling nearly always entails a chronological ordering to be enforced. Second, activities can be broken down into sub-activities; these represent sub-goals with respect to the purposes of the activity itself. Opposing PREREQUISITE/POSTREQUISITE links connect frames possessing a logical ordering. The values of a Prerequisite slot name frames which must immediately precede it. Analogous SUB/SUPER links connect frames subordinate one to another. A plan is a group of frames connected by these pointers. These implement a procedural net in the style of Sacerdoti [75], which served to unify the ideas of ABSTRIPS and NOAH as schemes for representing planning knowledge.

An example of using a plan is illustrated in the scenario. NUDGE's response R2 alludes to a PA progress report, whose frame contains the following planning links.



Interconnections describing its subgoals and the order in which they must be accomplished permit the creation of an instance mirroring this structure which satisfies the request. At R2 in the scenario, NUDGE makes a point of scheduling the preparation of a written progress report for Bruce which is clearly something to be accomplished before the newly scheduled meeting with Ira. The generic frame for PA-MEETING has a PREREQUISITE slot containing a requirement for this and an If-needed procedure to accomplish it.

```
(PA-MEETING
  (PREREQUISITE ($REQUIRE    ( (AKO REPORT) ))
                ($IF-NEEDED ( (INSTANTIATE-AS-REQUIRED) )))
  ...)
```

Frames and Knowledge. Frame systems have proved a convenient representation for knowledge that naturally falls into a taxonomy of successively more general categories. Individual frames are convenient for representing concepts that have multi-dimensional descriptions which may be potentially incomplete or inconsistent. However, the limits of frames as a representational scheme are not yet clearly understood. We plan extensions into different domains to understand these limitations.

6. BARGAINING BETWEEN GOALS

NUDGE translates an ill-defined, under-specified scheduling request into a complete specification, represented by the frame gestalt. This gestalt becomes the input to a scheduling program, BARGAIN, that seeks the best time for the requested activity if the desired time is unavailable. Other traditional scheduling programs could be employed as the gestalt is a complete and formal request. We use BARGAIN since it improves upon traditional decision analysis programs by incorporating AI techniques to control the search process.

BARGAIN is power-based in the sense that its competence is predicated on efficient search. It engages in a best-first search, as controlled by a static evaluation

function that measures (1) the number of violated preferences, (2) their respective utilities and (3) the number of remaining conflicts. BARGAIN was originally designed by Goldstein [75] and implemented in CONNIVER by F. Kern [75].

BARGAIN employs a set of 8 search operators which constitute debugging strategies for time conflicts. One set are "resource-driven", i.e. they are experts on the physics of time and eliminate a conflict by altering the duration, interrupting, sharing or moving the event. The second set are "purpose-driven" and go outside the time domain to examine the topic of the meeting and alternative methods for accomplishing it. An application of any one of these techniques produces a new calendar with the conflict resolved, and possibly new conflicts introduced. Each strategy has a cost associated with it. BARGAIN halts when it has found the best sequence of debugging strategies that generate a conflict free calendar within various computational constraints.

The applicability of a search operator -- especially the purpose-driven kind -- can depend on the overall knowledge context. Hence, the power-based approach benefits from some heterarchy with the preceding knowledge-based phase. A given search operator may ask the frame system whether it applies. For example, a strategy to change participants must rely on knowledge of available candidates and the goals of the activity for suggesting suitable replacements.

The relative preference for different scheduling strategies is controlled by specific assertions in the HOW slot, which contains the names of strategies applicable to the activity in which it appears. For example, PA meetings can be postponed as a last resort and only reluctantly interrupted; as can be seen in this excerpt from the PA-MEETING frame.

```
(PA-MEETING
    (HOW    ($DEFAULT ( (POSTPONE (MAXIMUM: 2)) (UTILITY: HIGH) ))
                      ( (INTERRUPT (UTILITY: HIGH) ))))
    ...)
```

Our approach to power-based scheduling parallels the conservative development of the knowledge-based component in that the well-understood techniques of decision analysis have been augmented only as required. This augmentation has involved applying AI search techniques to improve the efficiency with which a best compromise is found.

7. CONCLUSIONS

(1) FRL-0 provides a simple, but powerful representation technology. Frames are generalized property list, sharing much of the simplicity of traditional attribute/value representation schemes. Yet the addition of a few capabilities -- comments, constraints, defaults, procedural attachment, inheritance -- provides a great deal more power and economy in the representation. KRL and OWL are more ambitious and more complex, and may well apply to contexts in which FRL-0 proves insufficient. But this remains to be seen. We plan further experiments with FRL-0 to identify its strengths and weaknesses.

(2) Whether FRL-0 or some other AI language is employed, our experience with the nature of informal requests, the issues raised by multiple inheritance paths, the interaction between a search program and a rich knowledge base, and the epistemology of information transfer activities, time, place and people will surely be relevant to the design of knowledge-based AI programs.

(3) FRL is an experiment in the utility of the frames. Our experience is that clustering the answers to common questions about the frame structure for a concept provides a useful representation paradigm. The gestalt derived from this frame structure supplies missing information similar to that generated by competent human schedulers to handle informal requests.

(4) The entire system can be viewed from another perspective. Since a frame's behavior is actually governed largely by the attached procedures, it can be viewed as an accessing scheme to the underlying procedural knowledge. Thus, frames implement goal-directed invocation (as in PLANNER), but with pattern matching replaced by the more general process of frame instantiation.

(5) NUDGE is a step towards an AI system with common sense. By generating a complete frame gestalt, the system minimizes the possibility of overlooking obvious alternatives. Defaults, preferences and requirements allow much to remain unsaid. A tolerance for minor inconsistencies is a benchmark of a robust knowledge system.

(6) NUDGE and BARGAIN are steps towards the creation of an automated office. Given the enormous information flow in a modern office, this is an important area for applied

AI research.

8. BIBLIOGRAPHY

Balzer, R. 1974. "Human Use of World Knowledge," ISI/RR-73-7.

Bobrow, D.G., R. M. Kaplan, M. Kay, D. Norman, H. Thompson, T. Winograd. 1976. "GUS, A Frame-Driven Dialog System" Xerox Research Center, Palo Alto, Cal.

Bobrow, D.G. & Winograd, T. 1976. An Overview of KRL, a Knowledge Representation Language. Xerox Research Center, Palo Alto, Cal.

Bullwinkle, C. 1977. "The Semantic Component of PAL: The Personal Assistant Language Understanding Program", forthcoming Working Paper, MIT AI Laboratory.

Collins, A. and E. Warnock. 1974. Semantic Networks, Memo 2833, Bolt Beranek & Neuman.

Goldstein, I.P. 1974. Understanding Simple Picture Programs, MIT-AI TR 294.

Goldstein, I.P. 1975. "Bargaining Between Goals," IJCAI IV. Tbilisi.

Hewitt, C. 1969. "PLANNER: A Language for Proving Theorems in Robots", IJCAI I.

Kern, F. 1975. A Heuristic Approach to Scheduling. M.S. Thesis, MIT.

Marcus, M. 1976. "A Design for a Parser for English", Proc. of the ACM Conference, Houston, October 1976, pp. 62-68.

Martin, W.A. 1977. "A Theory of English Grammar." In A Computational Approach to Modern Linguistics: Theory and Implementation, (in preparation).

McDermott, D and Sussman, G. 1972. "The CONNIVER Reference Manual", MIT AI Memo 259.

Minsky, M. 1975. "A Framework for Representing Knowledge." In P.H. Winston(Ed.), The Psychology of Computer Vision, NY:McGraw-Hill.

Minsky, M. & Papert, S. 1974. Artificial Intelligence. University of Oregon.

Moore, J. and Newell, A. 1973. How Can MERLIN Understand? In L. Gregg (Ed.), Knowledge and Cognition, Lawrence Erlbaum Assoc., Potomac, MD.

Quillian, R. 1968. "Semantic Memory" in M. Minsky (ed.), Semantic Information Processing, MIT Press, Cambridge, Mass.

Sacerdoti, E. 1975. "The non-linear nature of plans." IJCAI IV.

Shank, R.C. 1973. "Identification of Conceptualizations Underlying Natural Language." In R.C. Shank & K.M. Colby (Ed.) Computer Models of Thought and Language, SF:Freeman.

Sussman, G.J. 1973. A Computational Model of Skill Acquisition, MIT-AI TR 297.

Tonge, F.M. 1963. "Summary of a Heuristic Line Balancing Procedure." In E. Feigenbaum & J. Feldman (Ed.), Computers and Thought. NY:McGraw-Hill.

Winograd, T. 1975. "Frame Representations and the Declarative/Procedural Controversy", in R. Bobrow and A. Collins (eds.), Representation and Understanding, NY Academic Press.