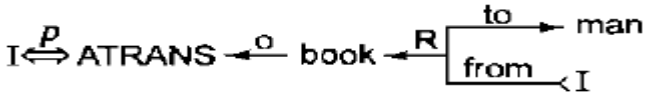
Strong slot and filter structures

CONCEPTUAL DEPENDENCY

Conceptual dependency (CD) is a theory of how to represent the kind of knowledge about events that is usually contained in natural language sentences. The goal is to represent the knowledge in a way that

- Facilitates drawing inferences from the sentences.
- Is independent of the language in which the sentences were originally stated.

Because of the two concerns just mentioned, the CD representation of a sentence is built not out of primitives corresponding to the words used in the sentence, but rather out of conceptual primitives that can be combined to form the meanings of words in any particular language. Unlike semantic nets, which provide only a structure into which nodes representing information at any level can be placed, conceptual dependency provides both a structure and a specific set of primitives, at a particular level of granularity; out of which representations of particular pieces of information can be constructed. As a simple example of the way knowledge is represented in CD, the event represented by the sentence **I gave the man a book** would be represented as:



Where the symbols have the following meaning:

- Arrows indicate direction of dependency.
- Double arrow indicates two way link between actor and action.
- p indicates past tense.

•ATRANS is one of the primitive acts used by the theory. It indicates transfer of possession.

- o indicates the object case relation.
- R indicates the recipient case relation.

In CD, representations of actions are built from a set of primitive acts. A typical set is the following:

ATRANS- Transfer of an abstract relationship (e.g., give)

PTRANS- Transfer of the physical location of an object (e.g., go)

PROPEL- Application of physical force to an object (e.g., push)

MOVE- Movement of a body part by its owner (e.g., kick)

GRASP- Grasping of an object by an actor (e.g., clutch)

INGEST- Ingestion of an object by an animal (e.g., eat)

EXPEL- Expulsion of something from the body of an animal (e.g., cry)

MTRANS- Transfer of mental information (e.g., tell)

MBUILD- Building new information out of old (e.g., decide)

SPEAK- Production of sounds (e.g., say)

ATTEND- Focusing of a sense organ toward a stimulus (e.g., listen)

A second set of CD building blocks is the set of allowable dependencies among the conceptualizations described in a sentence. There are four primitive conceptual categories from which dependency structures can be built. These are:

ACTs- Actions, PPs- Objects (picture producers), AAs- Modifiers of actions (action aiders), PAs- Modifiers of PPs (picture aiders)

In addition, dependency structures are themselves conceptualizations and can serve as components of larger dependency structures. The dependencies among conceptualizations correspond to semantic relations among the underlying concepts. Figure 10.2 lists the most important ones allowed by CD. The first column contains the rules; the second contains examples of their-use and the third contains an English version of each example. The rules shown in the Fig. can be interpreted as follows:

1. Rule 1 describes the relationship between an actor and the event he or she causes. This is a two-way dependency since neither actor nor event can he considered primary. The letter p above the dependency link indicates past tense.
2. Rule 2 describes the relationship between a PP and a PA that is being asserted to describe it. Many state descriptions, such as height, are represented in CD as numeric scales.
3. Rule 3 describes the relationship between two PPs, one of which belongs to the set defined by the other.
4. Rule 4 describes the relationship between a PP and an attribute that has already been predicated of it. The direction of the arrow is toward the PP being described.
5. Rule 5 describes the relationship between two PPs, one of which provides a particular kind of information about the other. The three most common types of information to be provided in this way are possession (shown as POSS-BY), location (shown as LOC),

and physical containment (shown as CONT). The direction of the arrow is again toward the concept being described.

6. Rule 6 describes the relationship between an ACT and the PP that is the object of that ACT. The direction of the arrow is toward the ACT since the context of the specific ACT determines the meaning of the object relation.
7. Rule 7 describes the relationship between an ACT and the source and the recipient of the ACT.
8. Rule 8 describes the relationship between an ACT and the instrument with which it is performed. The instrument must always be a full conceptualization (i.e it must contain an ACT), not just a single physical object.

Fig. 10.2 *The Dependencies of CD*

9. Rule 9 describes the relationship between an ACT and its physical source and destination.
10. Rule 10 represents the relationship between a PP and a state in which it started and another in which it ended.
11. Rule 11 describes the relationship between one conceptualization and another that causes it. The arrows indicate dependency of one conceptualization on another and so point in the opposite direction of the implication arrows. The two forms of the rule describe the cause of an action and the cause of a state change.
12. Rule 12 describes the relationship between a conceptualization and the time at which the event it describes occurred.
13. Rule 13 describes the relationship between one conceptualization and another that is the time of the first.
14. Rule 14 describes the relationship between a conceptualization and the place at which it occurred.

Conceptualizations representing events can be modified in a variety of ways to supply information normally indicated in language by the tense, mood, or aspect of a verb form. The set of conceptual tenses includes:

p- Past, f- Future, t- Transition, t_s- Start transition, t_f -Finished transition, k- Continuing, ? – Interrogative, / - Negative, Nil -Present, Delta -Timeless, c- conditional

As an example of the use of these tenses, consider the CD representation shown in Fig. 10.3 of the sentence **Since smoking can kill you, I stopped.**

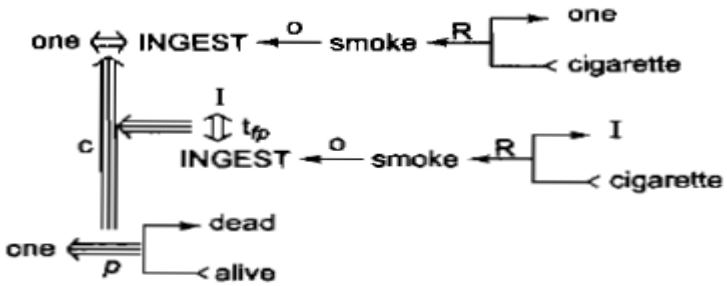


Fig. 10.3 *Using Conceptual Tenses*

The vertical causality link indicates that smoking kills one. Since it is marked c, however, we know only that smoking can kill one, not that it necessarily does. The horizontal causality link indicates that it is that first causality that made me stop smoking. The qualification t_{fp} attached

to the dependency between I and INGEST indicates that the smoking {an instance of INGESTING} has stopped and that the stopping happened in the past.

There are three important ways in which representing knowledge using the conceptual dependency model facilitates reasoning with the knowledge:

1. Fewer inference rules are needed than would be required if knowledge were not broken down into primitives.
2. Many inferences ate already contained in the representation itself.
3. The initial structure that is built to represent the information contained in one sentence will have holes that need to be filled. These holes can serve as an attention for the program that must understand ensuing sentences.

The **first** argument in favor of representing knowledge in terms of CD primitives is that using the primitives makes it easier to describe the inference rules by which the knowledge can be manipulated. Rules need only be represented once for each primitive ACT rather than once for every word that describes that ACT. For example, all of the following verbs involve a transfer of ownership of an object: **Give, Take, Steal, Donate**. If any of them occurs, then inferences about who now has the object and who once had the object may be important. In a CD representation, those possible inferences can be stated once and associated with the primitive ACT ATRANS.

A **second** argument is that to construct it, we must use not only the information that is stated explicitly in a sentence but also a set of inference rules associated with the specific information. Having applied these: rules once, we store these results as part of the representation and they can be used repeatedly without the rules being reapplied. For example, consider the sentence **Bill threatened John with a broken nose.**

The CD representation of the information contained in this sentence is shown in Fig. 10.4. It says that Bill informed John that he (Bill) will do something to break John's nose. Bill did this so that John will believe that if he (John) does some other thing (different from what Bill will do to break his nose), then Bill will break John's nose. In this representation, the word "believe" has been used to simplify the example. But the idea behind believe can be represented in CD as an MTRANS of a fact into John's memory. The actions do1 and do2, are dummy placeholders that refer to some as yet unspecified actions.

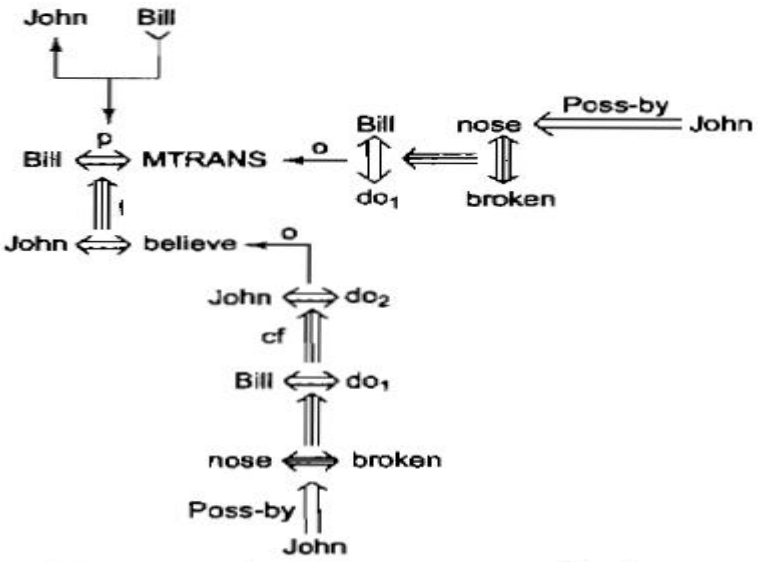


Fig. 10.4 *The CD Representation of a Threat*

A **third** argument is that unspecified elements of the representation of one piece of information can be used as a focus for the understanding of later events as they are encountered. So, for example, after hearing that **Bill threatened John with a broken nose,** we might expect to find out what action Bill was trying to prevent John from performing. That action could then be substituted for the dummy action represented in Fig. 10.4 as do2. The presence of such dummy objects provides clues as to what other events or objects are important for the understanding of the known event.

There is also argument, **against** the use of CD as a representation formalism. It requires that all knowledge be decomposed into fairly low-level primitives. Another **difficulty** is that it is only a theory of the representation of events. But to represent all the information that a complex program may need, it must be able to represent other things besides events. Thus, although there are several arguments in favor of the use of CD as a model for representing events, it is not always completely appropriate to do so, and it may be worthwhile to seek out higher-level primitives.

SCRIPTS

CD is a mechanism for representing and reasoning about events. But rarely do events occur in isolation. Script is a mechanism for representing knowledge about common sequences of events. It is a structure that describes a stereotyped sequence of events in a particular context. A script consists of a set of slots. Associated with each slot may be some information about what kinds of values it may contain as well as a default value to be used if no other information is available. Figure 10.5 shows part of a typical script, the restaurant script.

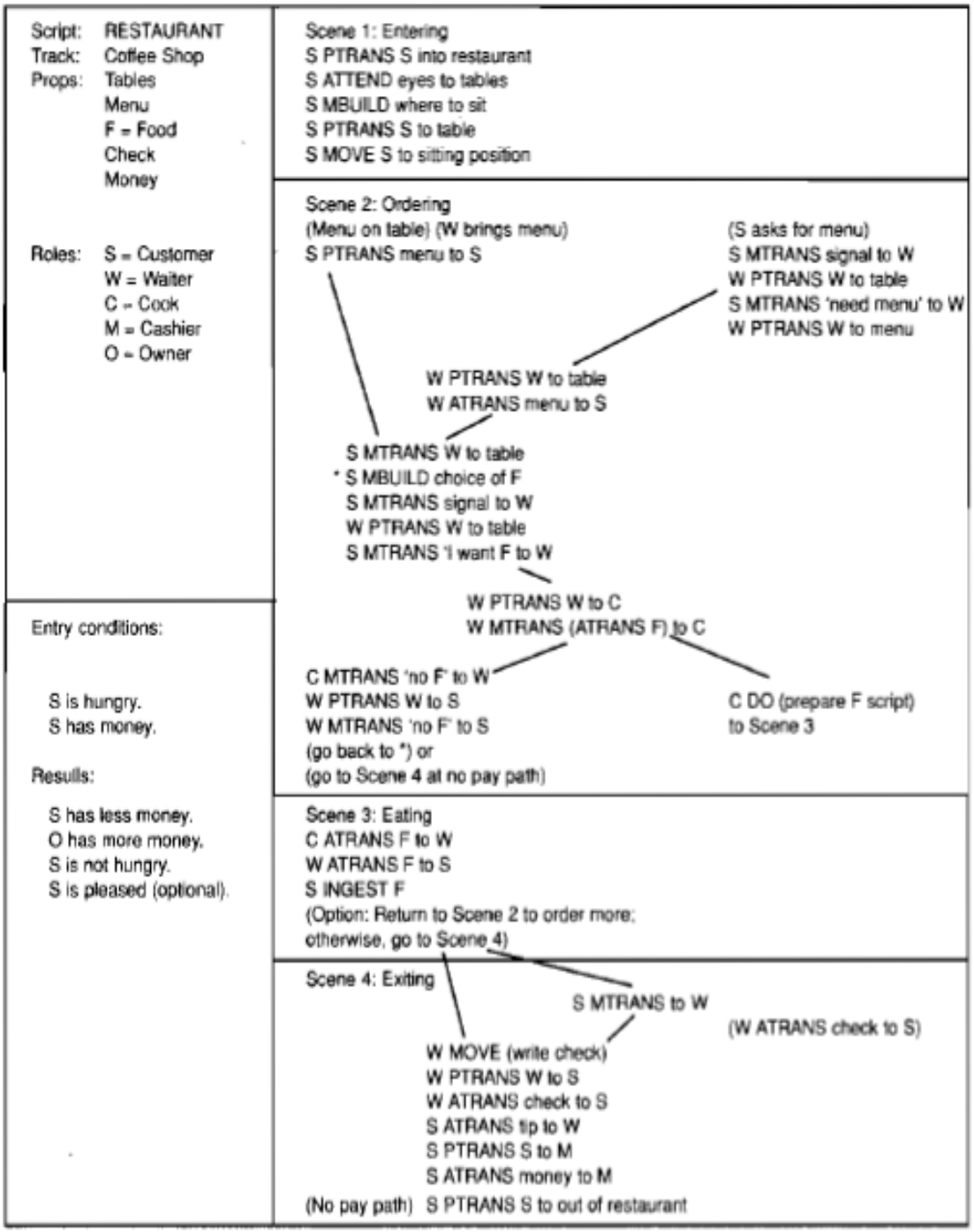


Fig. 10.5 *The Restaurant Script*

It illustrates the important components of a script:

Entry conditions: Conditions that must, in general, be satisfied before the events described in the script can occur.

Result: Conditions that will, in general, be true after the events described in the script have occurred.

Props: Slots representing objects that are involved in the events described in the script. The presence of these objects can be inferred even if they are not mentioned explicitly.

Roles: Slots representing people who are involved in the events described in the script. The presence of these people, too can be inferred even if they are not mentioned explicitly. If specific individuals are mentioned, they can be inserted into the appropriate slots.

Track: The specific variation on a more general pattern that is represented by this particular script. Different tracks of the same script will share many but not all components.

Scenes: The actual sequences of events that occur. The events are represented in conceptual dependency formalism.

Scripts are useful because, in the real world there are patterns to the occurrence of events. These patterns arise because of causal relationships between events. Agents will perform one action so that they will then be able to perform another. The events described in a script form a giant causal chain. The beginning of the chain is the set of entry conditions which enable the first events of the script to occur. The end of the chain is the set of results which may enable later events or event sequences to occur. Within the chain, events are connected both to earlier events that make them possible and to later events that they enable.

If a particular script is known to be appropriate in a given situation, then it can be very useful in predicting the occurrence of events that were not explicitly mentioned. Scripts can also be useful by indicating how events that were mentioned relate to each other. But before a particular script can be applied, it must be activated (i.e., it must be selected as appropriate to the current situation). There are two ways in which it may be useful to activate a script, depending on how important the script is likely to be:

- For fleeting scripts (ones that are mentioned briefly and may be referred to again but are not central to the situation), it may be sufficient merely to store a pointer to the script so that it can be accessed later if necessary. This would be an appropriate strategy to take with respect to the restaurant script when confronted with a story such as Susan passed her favorite restaurant on her way to the museum. She really enjoyed the new Picasso exhibit.
- For non-fleeting scripts it is appropriate to activate the script fully and to attempt to fill in its slots with particular objects and people involved in the current situation.

Once a script has been activated, there are, a variety of ways in which it can be useful in interpreting a particular situation. The most important of these is the ability to predict events that have not explicitly been observed. Suppose, for example, that you are told the following story: **John went out to a restaurant last night He ordered steak. When he paid for it, he noticed that he was running out of money. He hurried home since it had started to rain.** If you were then asked the question: Did John eat dinner last night? You would almost certainly respond that he did, even though you were not told so explicitly. Since all of the events in the story correspond to the sequence of events predicted by the script, the program could infer that the entire sequence predicted by the script occurred normally. Thus, it could conclude, in particular, that John ate.

A **second** important use of scripts is to provide a way of building a single coherent interpretation from a collection of observations. Thus it provides information about how events are related to each other. Consider, for example, the following story: **Susan went out to lunch. She sat down at a table and called the waitress. The waitress brought her a menu and she ordered a hamburger.** Now consider the question: Why did the waitress bring Susan a menu? The script provides two possible answers to that question:

•Because Susan asked her to. (This answer is gotten by going backward in the causal chain to find out what caused her to do it.)

•So that Susan could decide what she wanted to eat. ('This answer is gotten by going forward in the causal chain to find out what event her action enables.)

A **third** way in which a script is useful is that it focuses attention on unusual events. Consider the following story: **John went to a restaurant. He was shown to his table. He ordered a large steak. He sat there and waited for a long tune. He got mad and left.** John did not get mad because he was shown to his table. He did get mad because he had to wait to be served. Once the typical sequence of events is interrupted, the script can no longer be used to predict other events. So, for example, in this story, we should not infer that John paid his bill. But we can infer that he saw a menu, since reading the menu would have occurred before the interruption.

So scripts are less general structures than are frames, and so are not suitable for representing all kinds of knowledge. However they can be very effective for representing the specific kinds of knowledge for which they were designed.