Natural Language Processing II (SC674)

Prof. Feng Zhiwei

Ch6. Some Important Grammars in NLP

6.1 Dependency Grammar (DG)

The lexical dependency information is important in modeling the lexical constituents that heads place on their arguments or modifiers.

An important class of grammar formalisms is based purely on this lexical dependency. In these dependency grammars, the constituents and phrase-structure rules do not play any fundamental role. The syntactic structure of a sentence is described purely in terms of words and binary semantic or syntactic relation between these words.

Dependency grammar war proposed by Desniere (1959) and Davis Hays separately. The term “dependency’ was first used by David Hays.

In a lexicalized Phrase-structure tree (P-tree), if we heighten the level of head to its highest mother node, then we may get a dependency tree (D-tree). For example

P-tree:

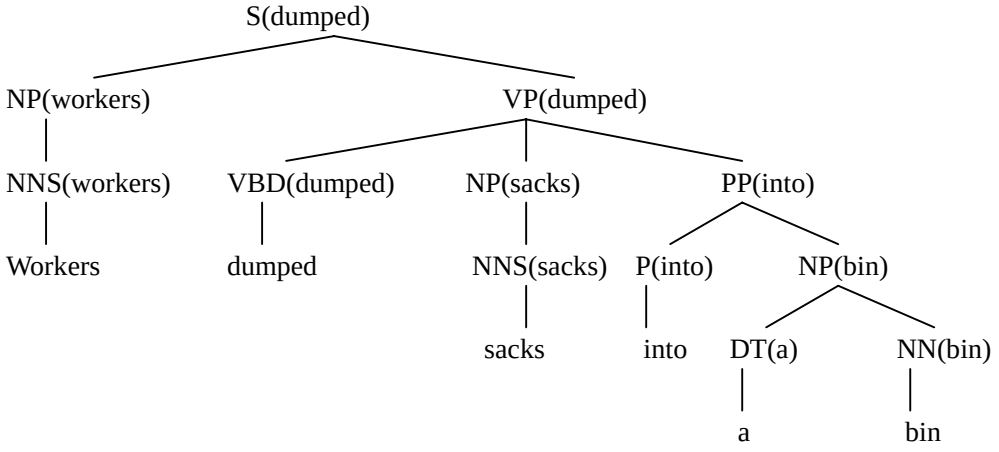
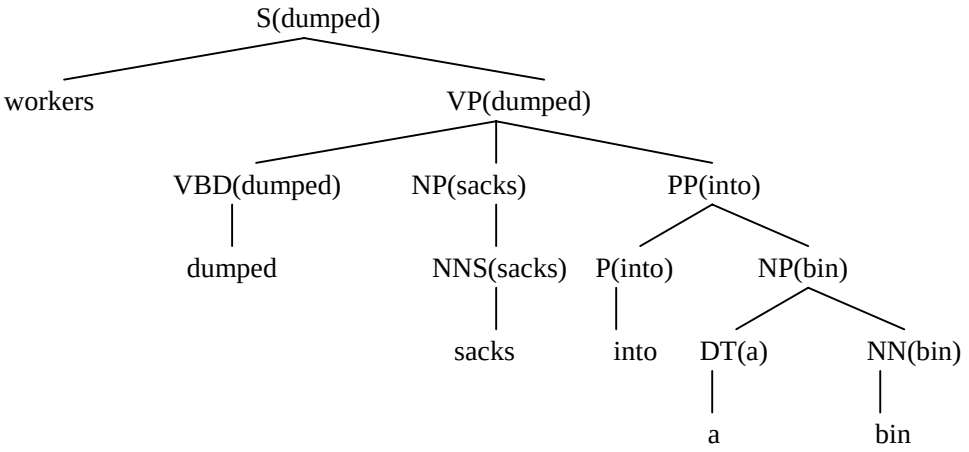
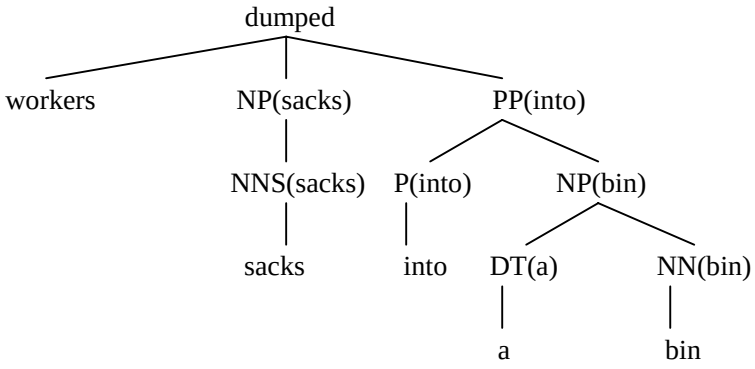


Fig. Lexicalized P-tree

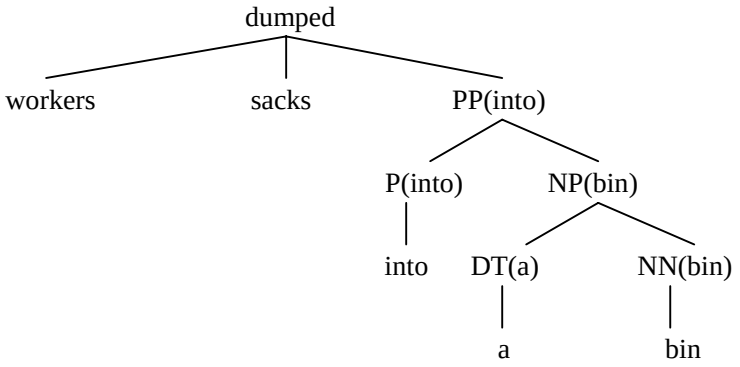
Step 1: heighten head “workers”:



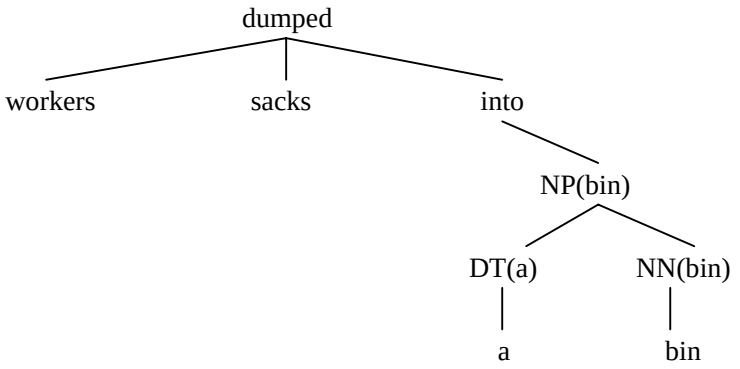
Step 2: Heighten head “dumped”:



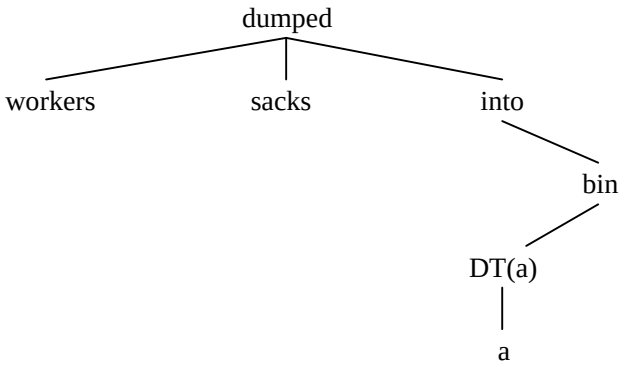
Step 3: heighten head “sacks”:



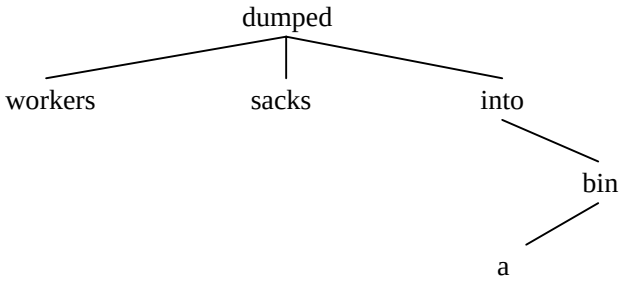
Step 4: heighten head “into”:



Step 5: heighten head “bin”::



Step 6: heighten head “a”:



Then we can obtain following D-tree:

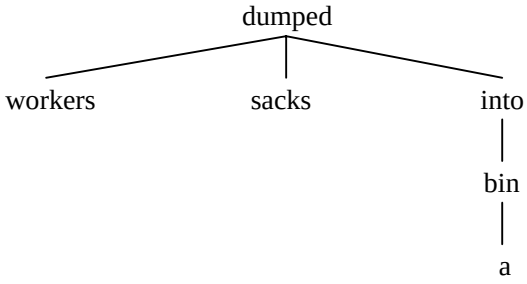


Fig. D-tree

In D-tree, there is no non-terminal of phrasal node. All the nodes are terminal. They are all concrete words in the sentence. Each link in the D-tree holds between two lexical nodes.

The link can express the relation between the nodes. Following is the relation used in Jaervinen and Tapanainen (1997) for English description:

Dependency Description:

Subj	syntactic subject
Obj	direct object (incl. Sentential complements)
Dat	indirect object
Pcomp	complement of a preposition
Comp	predicate nominals (complements of copulas)
Tmp	temporal adverbial
Loc	location adverbial
Attr	premodifying (attributive) nominals (genitives, etc.)
Mod	nominal postmodifiers 9preposition phrases, etc.)

Folowing is a D-trre with this description fro sentence “I gave him my address” :

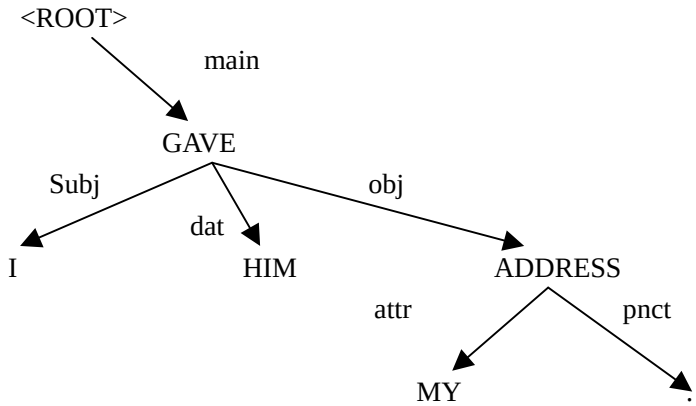
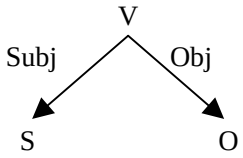


Fig. A sample D-tree using dependency description

The dependency grammar researchers argue that one of the main advantages of Dependency grammar is their ability to handle languages with relatively free word order. For the free word order languages, S-O-V, S-V-O have same dependency relation, but different word order, a CFG grammar would need a separate rule for each word order in the parse tree. A dependency grammar would just have one link type representing this kind relation:



Thus a dependency grammar abstracts away from word-order variation, representing only the information that is necessary for the parse.

6.2 Category Grammar (CG)

The Category Grammar (CG) was first proposed by Adjukiewicz (1935), and modified by Bar-Hillel (1953), Lambek (1958), Dowty (1979).

6.2.1 Syntactic types:

The English string “poor John sleeps” can be analyzed in the following way: “poor” is an adjective (Adj), “John” is a noun (N), “sleep’ is a verb (V), -s is a morpheme added to a verb “sleep”. “poor John” is a NP (adj + N), sleeps is a VP, and NP and Vp can form a sentence “Poor John sleeps”. It is customary method.

It we take n to represent noun category, take S to represent sentence category, then other categories can be represented by combination of n and S. This kind of category is called as syntactic type.

According to this notation, “John” will belong to the type n, “poor” to n/n (read n over n) , “sleeps” to n\S (read n under S), where n is to be interpreted as the category of namelike string, n/n as the category of those strings that with an n to their right form a string belonging to the same category n, and n\S as the category of those string that with an n to their left form a string belonging to the category of sentence S. That the string “poor John sleeps” is a sentence can now be tested mechanically, without recourse to any syntactic statements, by using something like ordinary multiplication of fraction on the index sequence corresponding to the given string (n/n)n(n\S).

Every word can belong to a syntactic type according to its function in the sentence. If we have syntactic type n and S, then other syntactic types can be derived by using following notation rules:

- if we have a word B, the syntactic type of its direct right word C is γ, and the syntactic type of the string BC is β, then the syntactic type of word B must be β/γ;
- if we have a word B, the syntactic type of its direct left word A is α, and the syntactic type of the string AB is β, then the syntactic type of word B must be α\β;
- if we have a word B, the syntactic type of its direct left word A is α, and the syntactic type of its direct right word C is γ, and the syntactic type of the string ABC is β, then the syntactic type of word B must be α\

β / γ.

The syntactic type of English word is as follows:

The syntactic type of John is n;

The word “poor” in “poor John”, the syntactic type of its right word “John” is n, the function of string “poor John” is same as the syntactic type n, so its syntactic type must be n/n.

The word “works” in “John works”, the syntactic type of its left word “John” is n, the function of string “John works” is same as the syntactic type S, so its syntactic type must be n\S.

The word “likes” in “John likes Jane”, the syntactic type of its left word “John” is n, the syntactic type of its right word ‘Jane” is also n, the function of string “John likes Jane” is same as the syntactic type S, so its syntactic type must be n\S/n.

The word “soundly” in “John slept soundly”, the syntactic type of its left word “slept” is n\S, the function of sequence “slept soundly” is same as the syntactic type n\S, so its syntactic type must be (n\S)\n\S.

The word “here” in “John works here”, the syntactic type of its left word “John works” is S, the function of string “John works here” is same as the syntactic type S, so its syntactic type must be S\S.

The word “never” in “John never works”, the syntactic type of its left word “John” is n, so the function of string “never works” is the syntactic type n\S, but the syntactic type of “works’ is n\S, so the syntactic type of “never” must be n\S/(n\S).

The word “for” in “John works for Jane”, the its syntactic function is similar as word “here” (S\S), but the syntactic type of its right word “Jane” is n, so its syntactic type must be S\S/n.

The word “and” in “John works and Jane rests”, the syntactic function is connect two clauses as one sentence, so its syntactic type must be S\S/S.

The syntactic types of English POS is in following table:

	Word	Syntactic Type	POS
(1)	John	n	noun
(2)	poor	n/n	adjective
(3)	works	n/S	intransitive verb
(4)	likes	n\S/n	transitive verb
(5)	soundly	(n\S)\n\S	adverb
(6)	here	S\S	adverb
(7)	never	n\S/(n\S)	adverb
(8)	for	S\S/n	preposition
(9)	and	S\S/S	conjunction

The syntactic types correspond approximately to the traditional part of speech (POS). Thus, in (1) John is a noun, in (2) poor is an adjective, in(3)works is an intransitive verb, in (4) likes is a transitive verb, in (5) soundly, in (6) here, in (7) never are different adverb with different functions, in (8) for is a

preposition, in (9) and is a conjunction. The syntactic types of English can be represented by n and S. n and s are atomic categories. Other syntactic types which was derived by n and s are complex categories. In logical aspect, S represents a true-value proposition, and n represents the argument in this proposition, it is a very concise and simple representation for the sentence in natural language. All information ncan be derived from these categories which can represents the syntactic type of different words stored in dictionary. It is a typical lexicalism in NLP.

6.1.2 Syntactic Type Calculus:

The syntactic type calculus rules are as following:

If we have the category sequence "α, α\β/γ. γ", then we can replace it by β.

(α)(α\β/γ)(γ) → β

This rule can be divided by following two rules:

- Rule ①: Replace sequence "α, α\β" by β: (α)(α\β) → β
- Rule ②: Replace sequence "β/γ, γ" by β: (β/γ)(γ) → β

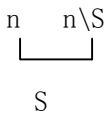
Based on these rules, in the string (n/n)n(n\S) (corresponding to sentence "poor John sleeps"), we may simplify the sequence (n/n)n, to the right and obtain n; the whole sequence becomes n(n\S); we may simplify to the left and we get the type S. That is, "poor John sleeps" is a well-formed English sentence.

(n/n) n (n\S) → n(n\S) → S

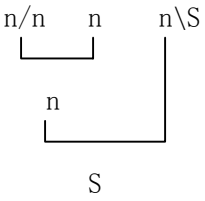
In the parse based category grammar, if we can obtain S after several syntactic type calculus through finite steps, then we can say: the input sentence is well-formed sentence.

Another examples:

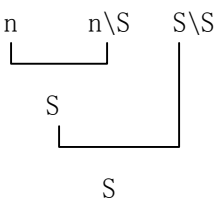
1: John works



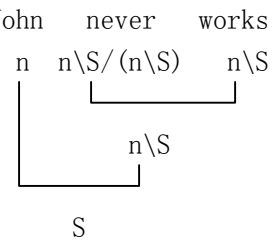
2: Poor John works



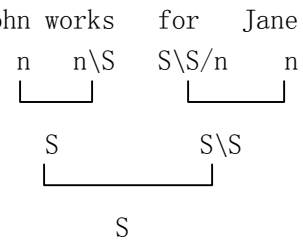
3: John works here



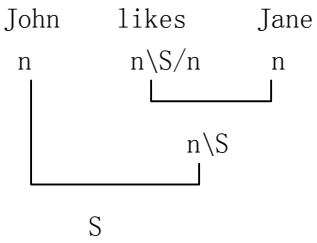
4:



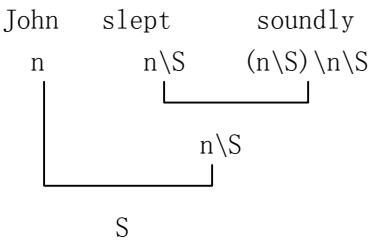
5:



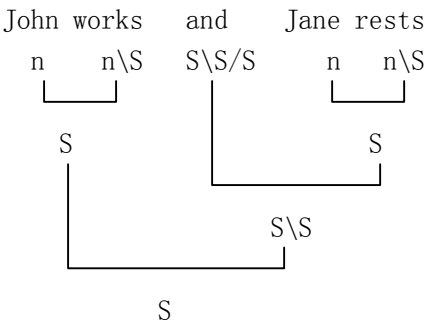
6:



7:



8:

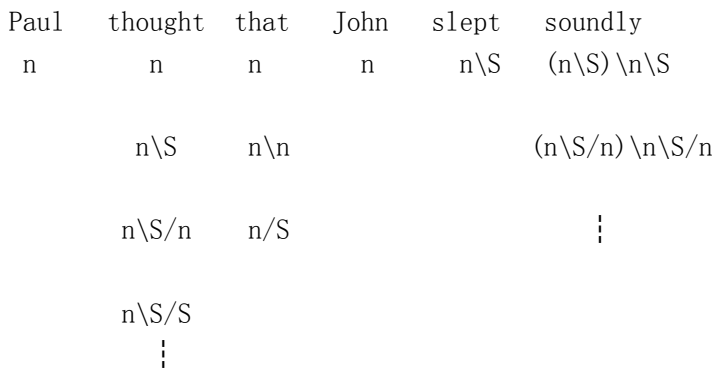


A word can belong to different categories. E.g., word “know”, in “John knows”, its category is n\S, in “John knows Jane”, its category is n\S/n. In the concrete calculus, we have to list all categories for the words including in the sentence.

E.g., We have the sentence:

Paul thought that John slept soundly

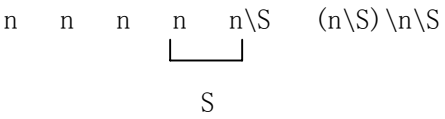
We list the categories of every words in the bottom of corresponding word:



We assume the categories of word “thought” are only 4 as above, the categories of word “soundly” are only 2 as above, then we can obtain 24 initiating category sequences: (1×4×3×1×1×2=24)。

Now we calculate these category sequences:

In category sequence “n n n n n\S (n\S)\n\S”,

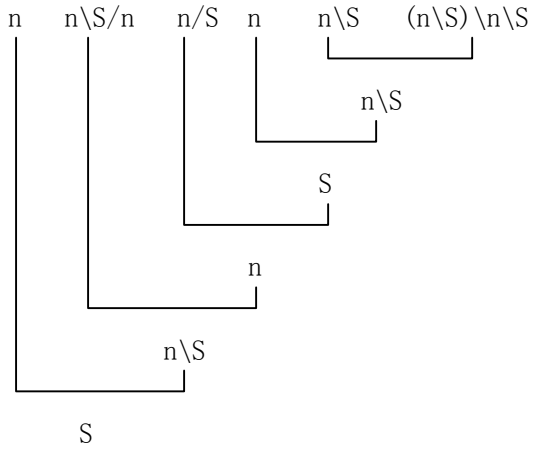


We use rule ① to category 4 and 5, then we obtain “n n n S (n\S)\n\S. For this sequence, any rule cannot be further used.

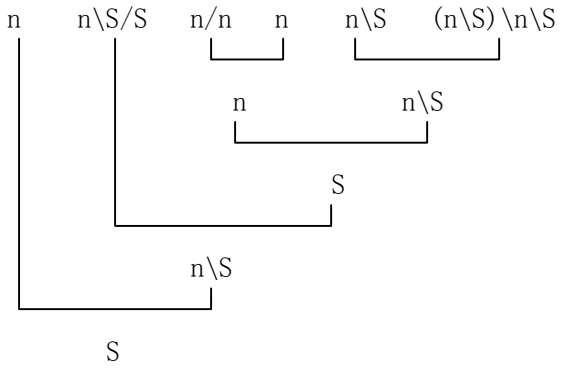
We may also use rule ① to category 5 and 6, then we obtain “n n n n n\S”. For this sequence, any rule cannot be further used.

In the last, we can obtain two results:

First result:



Second result:



Here we obtain two results. Every result represents different meaning:

First result means: "Paul thought, John slept soundly". Here, "that" is the relative pronoun which introduces a clause "John slept soundly".

Second result means: "Paul thought, that John sleepy soundly". Here, "that" is demonstrative pronoun which modifies "John".

The results show that this sentence is ambiguous.

To distinguish between different forms such as "works" and "work", we might assign the type n* to all plural noun, such as "men", or "chairs". Thus we can have:

"men work"
n* n*\S
"poor men work"
n*/n* n* n*\S
"John work for men"
n n\S S\S/n* n*
"John likes girls"
n n\S/n* n*
"men like Jane"
n* n*\S/n n

By this means, we can calculate the sentence including plural form of noun.

In example 5 "John works for Jane", if we first calculate the sequence "works for", we shall meet the difficulty.

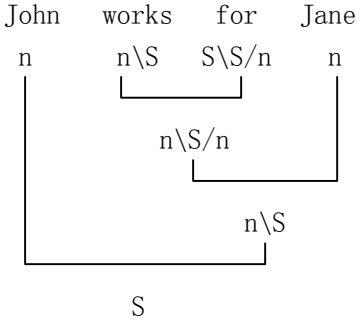
John works for Jane
n n\S S\S/n n
?

In this case, we cannot use any rule to deal with sequence "n\S S\S/n".

Lambek proposes following two rules:

③ $(\alpha \backslash \beta)(\beta \backslash \gamma) \rightarrow \alpha \backslash \gamma$
④ $(\alpha / \beta)(\beta / \gamma) \rightarrow \alpha / \gamma$

By these rules, we can calculate the above sequence:



In this case, the rules of syntactic type calculus become to following 4 rules:

① (α)(α\β) → β

② (β/γ)(γ) → β

③ (α\β)(β\γ) → α\γ

④ (α/β)(β/γ) → α/γ

6.2.3 English verb processing in category grammar:

Some sentences will escape our above net of category grammar. We adopt following primitive types:

S: complete declarative sentence;

n: name;

i: infinitive of intransitive verb;

p: present participle of intransitive verb;.

q: past participle of intransitive verb.

A number of key sentences containing different forms of intransitive "work" will illustrate our choice of types:

"must" in "John must work": "work" is intransitive verb, its category is i, the left of "must" is n, the right of "must" is i, and the function of string "John must work" is same as the sentence, so the category of "must" is n\S/i.

"is" in "John is working": "working" is the present participle of intransitive verb, its category is p, the left of "is" is n, the right of "is" is p, and the function of string "John is working" is same as the sentence, so the category of "it" is n\S/p.

"has" in "John has worked": "worked" is past participle of intransitive verb, its category is q, the left of "has" is n, the right of "has" is q, and the function of string "John has worked" is same as the sentence, so the category of "has" is n\S/q.

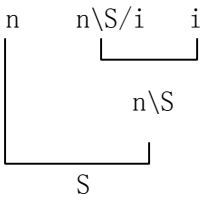
In this case, the category of "must work, is working, has worked" all are n\S, it function is similar as a bare-stem verb 'work".

"be" in "John must be working": "working" is present participle of intransitive verb, its category is p, the function of string "be working" is same as the infinitive verb, so the category of "be" is i/p.

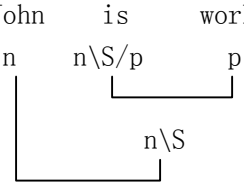
"been" in "John has been working": "working" is present participle of intransitive verb, its category is p, the function of string "been working" is same as the past participle of intransitive verb, so the category of "been" is q/p.

Some Examples for the calculus of intransitive verb:

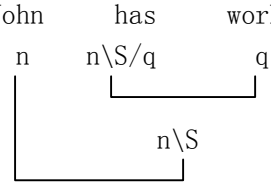
1. John must work



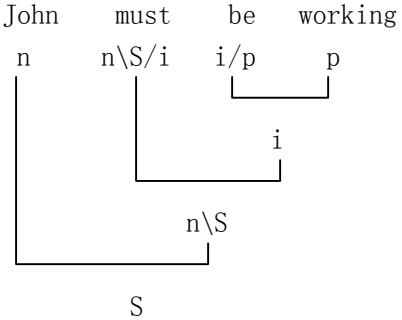
2. John is working



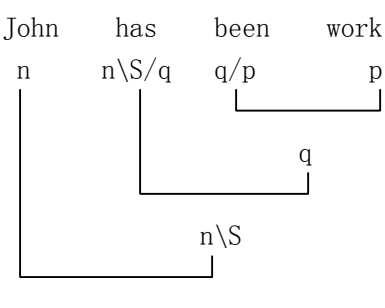
3. John has worked



4. John must be working



5. John has been working



Then we consider a number of sentences containing different forms of the transitive verb "call":

"calls" in "John calls Jane": "calls" is present, singular, 3rd person form of the transitive verb "call", the left of "calls" is n, the right of "calls" is also n, and the function of string "John calls Jane" is same as the sentence, so the category of "calls" is n\S/n.

"call" in "John must call Jane": "call" is transitive verb, the right of "call" is n, and the function of string "call Jane" is same as an infinitive of transitive

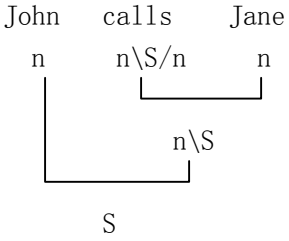
verb i, so the category of "call" is i/n.

"calling" in "John is calling Jane": the right of "calling" is n, and the function of string "calling Jane" is same as an present participle of transitive verb p, so the category of "calling" is p/n.

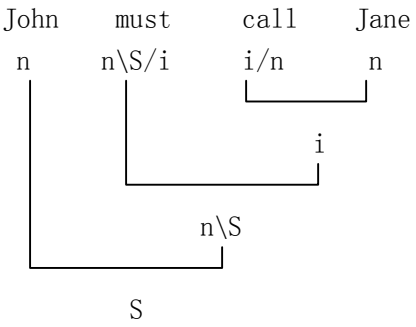
"called" in "John has called Jane": the right of "called" is n, and the function of string "called Jane" is same as a past participle of transitive verb q, so the category of "called" is q/n.

Some examples for calculus of transitive verb:

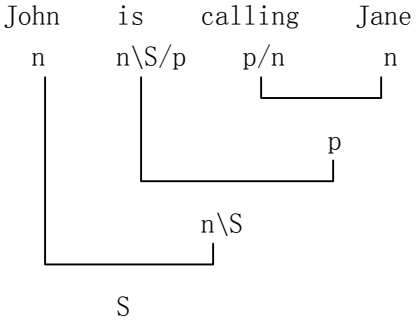
6.



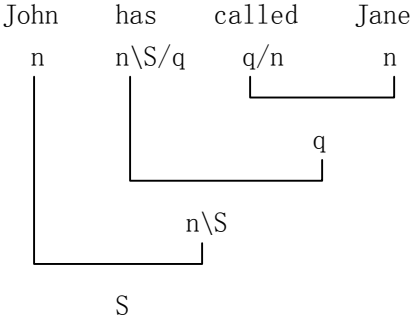
7.



8.



9.



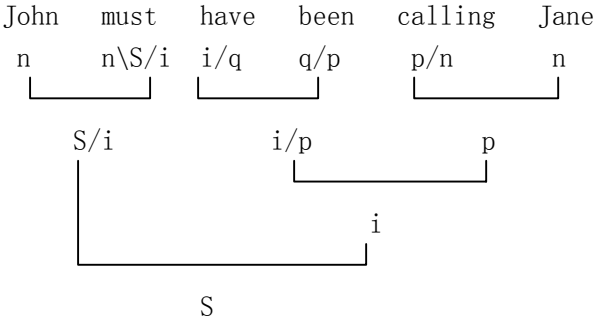
The resulting types are embodied in following table:

	Modal Auxiliary	Intransitive	transitive	Auxiliary	Progressive Auxiliary	Passive Auxiliary
Infinitive		work i	call i/n	have i/q	be i/p	be i/(q/n)
Present Participle		working p	calling p/n			being p/(q/n)
Past participle		worked q	called q/n		been q/p	been q/(q/n)
3rd person singular	must n\S/i	works n\q	calls n\S/n	has n\S/q	is n\S/p	is n\S/(q/n)

Table 1.

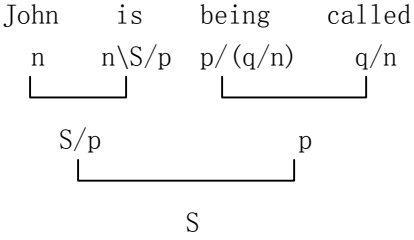
To illustrate calculation based on table 1., let us consider the following strings:

10.



In this sentence, "been" is progressive auxiliary, its category is q/p.

11.



In this sentence, "been" is passive auxiliary, its category is p/(q/n).

Table 1 is not complete. Principal omission are the following: doubly transitive verbs such as "give, appoint; verbs which relate nouns and adjectives, as in "the lunch tastes good"; first person form such as "am"; plural forms of the verb such as "are".

Table 1 is more extensive than it looks, since many entries are merely representative

samples. Thus ‘work″ represents all intransitive verbs, ‘‘calls″ represents all transitive verbs, and ‘‘must″ represents all modal auxiliaries such as ‘‘will, shall, can, may, would, should, could, might″.
Furthermore, the passive auxiliary ‘‘be″ nay often be replaced by ‘‘get″, and the progressive auxiliary ‘‘be″ seems to represent a large class of verbs, including at first sight ‘‘start, begin, keep, continue, stop, finish″.
Some of the forms appearing in the table 1 can also be different types. Thus ‘‘call″ may be a noun, and ‘‘have′ and ‘‘be′ may also appear as main verbs, as in ‘‘John must have lunch″ and ‘‘John must be good″.

The gaps in the first column of table 1 are because ‘‘must″ has no infinitive and the conceivable forms ‘‘musting″ (p/i) and ‘‘musted″ (q/I) do not exist. The gaps in the second and third columns of the table 1 exist because the auxiliary ‘‘have″ has no present and no past participle and the active auxiliary ‘‘be′ has no present participle. This is not quite correct if we consider ‘‘having″ in ‘‘having worked, John rested″ as a participle. However, the main verb ‘‘have′ and the passive auxiliary ‘‘be″ (as well as the main verb ‘‘be″) has a present participle, as is attested by the sentence: ‘‘John is having lunch; John has had lunch,; John is being called″. The corresponding types are having (p/n), had (q/n), being (p/(q/n)).

One can interpret table 1 as a kind of multiplication table as shown in table 2. The entries of table 2 are in fact obtained by multiplication according to rules ①-④:

Second / First	i/i	i	i/n	i/q	i/p	i/(q/n)
i/i	i/i	i	i/n	i/q	i/p	i/(q/n)
p/i	p/i	p	p/n	p/q	p/p	p/(q/n)
q/i	q/i	q	q/n	q/q	q/p	q/(q/n)
n\S/i	n\S/i	n\S	n\S/n	n\S/q	n\S/p	n\S/(q/n)

E.g. If first item is p/i, second item is i, because

(p/i)i → p

so the crossing point of p/i and i equals p。

If first item is q/i, second item is i/p, because

(q/i)(i/p) → q/p

So the crossing point of q/i and i/p equals q/p.

The value in the crossing point of table 2 can be extended in reversal direction. E.g. from (p/i)i → p in table 2, we can extend p to (p/i)i; from (q/i)(i/p) → q/p, we can extend q/p to (q/i)(i/p).

By this reversal extension, we can get some new knowledge on linguistic phenomena.

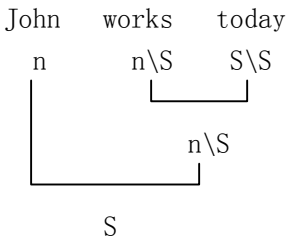
For example, the category of ‘‘works″ is n\S, we can extend n\S to (n\S/i)i,it

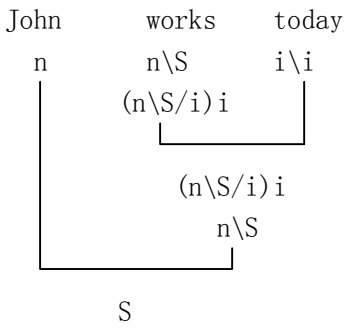
means that "works" can be regarded as two parts which category is respective n\S/i and i. In "does work", the category of "does" is n\S/i, and the category of "work" is i, therefore we can regard "does work" as the variation forms of "works". In fact, when "works" is in the question sentence, it must take the form "does work".

Compare, declarative sentence: John works

Question sentence: Does John work?

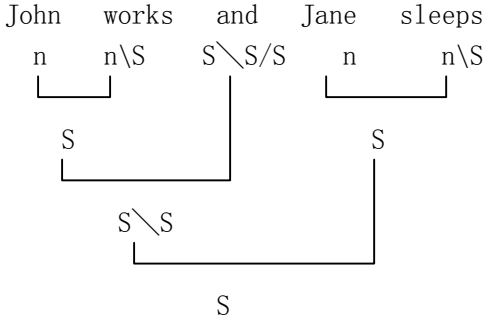
The category of today (adverb) is S\S. In "John works today", the categories of each word are respectively : "n, n\S and S\S", we have: n(n\S) → S, S(S\S) → S, so it is a well-formed sentence. However, we can also write the category of "today' as i\i, because if we in reversal direction extend n\S (the category of "works") to (n\S/i)i, and we take the category of "today' as i\i, then we shall have i(i\i) → i, and (n\S/i)i → n\S, therefore we have ((n\S/i)i)(i\i) → n\S, and n(n\S) → S. It means that "John works today' is a well-formed sentence. The calculation is as follows:



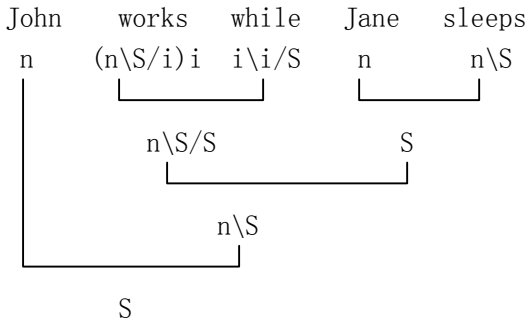


The new assignment also permits us distinguish between the types of coordinate and subordinate conjunctions.

E.g. In "John works and Jane sleeps", "and' is coordinate conjunction, its category is S\S/S, the calculation process is:



However, the category n\S of "works" can also be extended in reversal direction to (n\S/i)i, In this case, In "John works while Jane sleeps" the category of subordinate conjunction "while" becomes to i\i/S. Because after the reversal extension of works, the calculation process of this sentence is:



In this case, it can be regarded as a well-formed sentence.

Here, ((n\S/i)i)(i\i/S) → n\S/S, because we have

i(i\i/S) → i/S

(n\S/i)(i/S) → n\S/S

6.3 CD theory:

6.3.1 Fundamental of CD theory:

The Conceptual Dependency (CD) theory is as a representation for meaning of phrases and sentences. The three important ideas of CD theory are as follows:

- **The basic axiom of CD theory**

"For any two sentences that identical in meaning, regardless of language, there should be only one representation of that meaning in CD" (Schank and Abelson, Scripts, plans, goals, and understanding, Hillsdale, New York, Lawrence Eribaum, 1977)

Schank thus allies with the early machine translation concept of interlingua, or intermediate language, and has in fact done some machine translation research in conjunction with the story understanding project (SAM).

- **CD representations:**

The conceptual dependency representations are made up of a very small number of semantic primitives, which include primitive acts and primitive states (with associated attribute values).

--Primitive acts:

PTRANS: The transfer of the physical location of an object. E.g. For one to "go" is to PTRANS oneself. "Putting" an object somewhere is to PTRANS it to that place.

PROPEL: The application of a physical force to an object.

ATRANS: The transfer of an abstract relationship. E.g. To "give" is to ATRANS the relationship of possession or ownership.

MTRANS: The transfer of mental information from one person to another, or from one part of one's mind to another within a person. E.g. "telling" is an MTRANS between people; "seeing" is an MTRANS within a person.

MBUILD: The construction of new information from old. "Imaging", "inferring", and "deciding" are MBUILD.

We shall explain some primitive acts in detail:

MTRANS: There are hundreds of verbs whose main meaning element is MTRANS: “read, tell, see, hear, remember, forget, teach, suggest, promise” The important thing about each of these verbs is that they all have as their main focus, a transfer of information.

A naïve person thinks about when he hears a sentence using the word :”see”. We call the part of the brain (i.e. the mental location or MLOC) that receives incoming information the Conscious Processor (CP)

We also need an instrument action that specifies the ATTENDing, ATTEND’s object is any sense organ and its direction is some concrete object. Thus the meaning of “John saw Mary” is represented as:

Actor: John
Action: MTRANS
Object: image of Mary
Direction TO: MLOC (CP of John)
FROM: eyes of John

Instrument:
Actor: John
Action: ATTEND
Object: eyes
Direction TO: Mary
FROM: unknown

That is: “seeing” is MTRANS to the MLOC (CP) by means of ATTEND eyes.

Let us look at “reading” as in the sentence: “John read a book”.

This is represented as:

Actor: John
Action: MTRANS
Object: information in a book
Direction TO: CP of John
FROM: book

Instrument:
Actor: John
Action: ATTEND
Object: eyes
Direction TO: book
FROM: unknown

Notice that in this analysis “seeing” and “reading” are nearly identical. In fact. “seeing a book” and “reading a book” would be exactly identical except for one important thing” the project of MTRANS for “see X” is “the image of X”, The object of MTRANS for “read X” is “the information in X”.

Now we consider the verbs ‘tell” and “promise”. ”tell” is an MTRANS of information between people by speaking. It is very simple.

Now we consider the sentence “John promise to give Mary a book”.

Actor: John
Action: MTRANS
Object: Actor: John
Action: ATRANS
Object: book
Direction TO: Mary
FROM: John
Time: future
Direction TO: unknown
FROM: John

We notice that there is no instrument. The above sentence does not specify how the communication took place. “promise” can be made verbally, by letter, and by intermediaries. Since “promise” doesn’t say anything about instruments, we leave the instrument out.
There is an unknown TO direction. We might assume that Mary received this promise, and indeed this would be a good inference, but the sentence does not actually say so, and someone else could have in fact been the receiver of the promise.

PTRANS: “John went to New York”

Actor: John
Action: PTRANS
Object: John
Direction TO: New York
FROM: unknown

John took a plane to New York

Actor: John
Action: PTRANS
Object: John
Direction TO: New York
FROM: unknown

Instrument:

Actor: plane
Action: PROPEL
Object: plane
Direction TO: New York
FROM: unknown

and

Actor: John
Action: PTRANS

Object: John
Direction TO: plane
FROM: unknown

PROPEL: It underlies any action that involves force. Thus “to shoot” is to PROPEL bullets; “to kick” is to PROPEL one’s foot; “to throw” is to PROPEL an object. “Throw at” is PROPEL with the intention of “making contact”.

The rock hit the boy

Actor: unknown
Action: PROPEL
Object: rock
Direction TO: boy

CAUSE

State: contact
Object 1: rock
Object 2: boy

INGEST: “Drinking, eating, breathing” are all examples of INGEST.

E.g: “John drank a glass of milk.”

Actor: John
Action: INGEST
Object: milk
Direction TO: mouth of John
FROM: glass

Instrument:

Actor: John
Action: PTRANS
Object: glass containing milk
Direction TO: mouth of John
FROM: table

Instrument:

Action: John
Action: MOVE
Object: hand of John
Direction TO: glass
FROM: unknown

Instrument:

Actor: John
Action: GRASP
Object: glass of milk
Direction TO: hand of John

Here we see a case of infinite expansion of instruments. The analysis reads: “John drank the milk by getting the milk to his mouth by moving his hand to the milk and grasping the glass.”

E.g. “John shot up heroin”

Actor: John
Action: INGEST
Object: heroin
Direction TO: vein of John
FROM: hypodermic needle

“Hypodermic” is a needle for putting drugs into the body by picking the skin.

Instrument:

Actor: John
Action: PROPEL
Object: hypodermic needle
Direction TO: body-part of John
FROM: unknown

MBUILD: it is a process that receives an input either from memory or from outside, and produces an output which it places in the CP. Thus the representation for MBUILD will contain an “object IN” slot and an “object OUT” slot to account for the problem to be considered, and the solution reached by the MBUILDing process.

E.g. “John figured out where the money is”

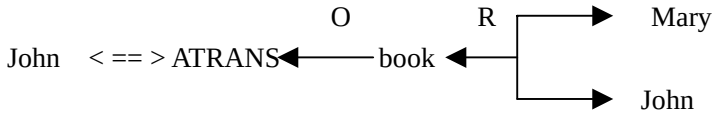
Actor: John
Action: MBUILD
Object IN: LOC (money) is place ?
OUT: LOC (money) is place X
Direction TO: CP
FROM: memory

E.g. “John wondered who ate the cheese”

Actor: John
Action: MBUILD
Object IN: Actor: ?
Action: INGEST
Object: cheese
Direction TO: mouth of?
FROM: unknown
OUT: Actor: none
Direction TO: CP

FROM: memory

In the most recent version of CD theory, Schank and Abelson included 11 these primitive acts.. Relations among concepts are called dependencies, and there is a fixed number of these, each represented graphically by a special kind of arrow. For example, the canonical representation of the sentence “John gives Mary a book“ is the following graph:



Where “John”, “book” and “Mary” are concept nodes, and the ATRANS node represents one of the primitive acts. The complicated, three-pointed arrow labeled “R” indicates a recipient-donor dependency between Mary and John and the book, since Mary got the book from John. The arrow labeled “O” indicates an “objective” dependency; that is, the book is the object of the ATRANS, since it is the thing being given. Dependency links may link concepts or other conceptual dependency networks.

--Primitive states:
The primitive states in CD theory include:

STATES:

Mary HEALTH (-10)	Mary is dead.
John MENTAL STATE (+10)	John is ecstatic. (in a state of very strong feeling of joy and happiness)
Vase PHYSICAL STATE (-10)	The vase is broken.

The number of primitive states is much large than the number of primitive actions. States and actions can be combined; for example, the sentence “John told Mary that Bill was happy” can be represented (in a newer arrowless representation) as

John MTRANS (Bill BE MENTAL-STATYE (5)) to Mary.

An important class of sentences involves “causal chains”, and Schank and Abelson and other their colleagues have worked out some rules about causality that apply to conceptual dependency theory. Five important rules are:

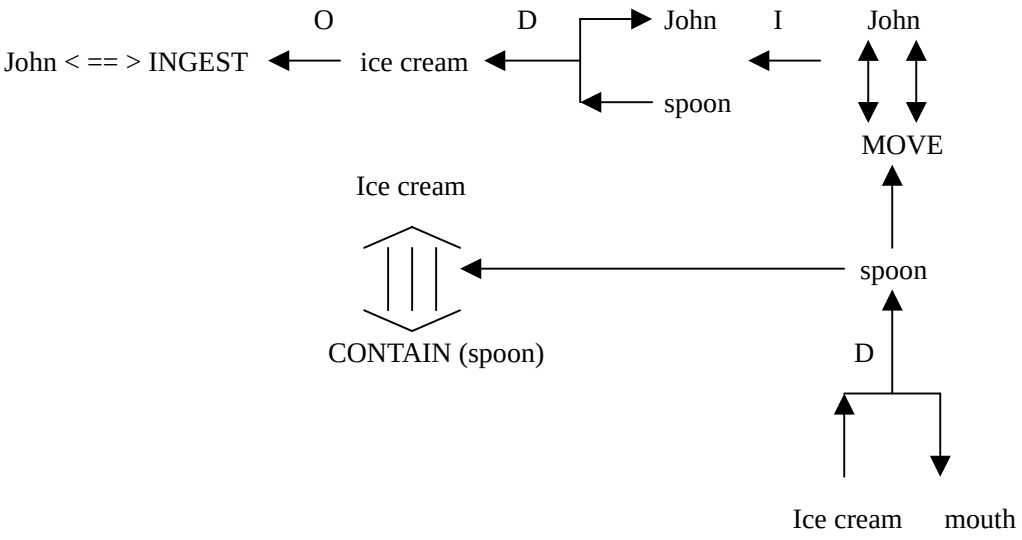
1. Actions can result in state changes.
2. States can enable actions.
3. States can disable actions.
4. States (or acts) can initiate mental events.
5. Mental events can be reasons for actions.

These are fundamental pieces of knowledge about the world, and CD includes a shorthand representation of each (and combinations of some) called “causal links”.

■ **Implicit must be made explicit**

The third important idea in CD theory is: "**Any information in a sentence that is implicit must be made explicit in the representation of the meaning of that sentence**" (Schank and Abelson, Scripts, plans, goals, and understanding, Hillsdale, New York, Lawrence Eribaum, 1977) This idea is the basis for much of the sophisticated inferential ability of MARGIE and the other systems.

It can be illustrated by the CD representation of the sentence "John eats the ice cream with a spoon":



Where the D and I arrows indicate direction (D) and instrument (I) dependencies, respectively. Note that in this example, "mouth" has entered the diagram as part of the conceptualization, even though it was not in the original sentence. This is a fundamental difference between conceptual dependency networks and the derivation tree that is produced in parsing a sentence. John's mouth as the recipient of the ice cream is inherent in the meaning of the sentence, whether it is expressed or not. In fact, the diagram can never be finished, because we could add such details as "John INGESTed the ice cream by TRANSing the ice cream on a spoon to his mouth, by TRANSing the spoon to the ice cream, by GRASPing the spoon, by MOVing his hand to the spoon, by MOVing his hand muscles," and so on. Such an analysis is known to both the speaker and the hearer of the sentence and normally would not need to be expanded.

For some tasks, like paraphrasing and question answering, CD representation has a number of advantages over more surface-oriented systems. In particular, sentence like

:

Shekespeare wrote Hamlet

And

The author of Hamlet was Shekespeare.

Which in some sense have the same meaning, map into the same CD structure.

Another important aspect of CD theory is its independence from syntax; in contrast with earlier work in paradigms of TG (transformational grammar) or PSG (phrase structure grammar), a

"parse" of a sentence in CD bears little relation to the syntactic structure. Schank (1975) also claims that CD has a certain amount of psychological validity, in that it reflects intuitive notions of human cognition.

6.3.2 The MARGIE system:

MARGIE system was a program developed by Roger Schank and his students at the Stanford AI Laboratory. Its intent was to provide an intuitive model of the process of natural language understanding. This system was programmed in LISP 1.6, is divided into three components.

- Conceptual analyzer:

  It was written by Christopher Riesbeck. This module took English sentences and converted them into an internal conceptual-dependency representation. It was done through a system of "requests", which are similar to "demons" or "production" in other AI systems. A request is essentially a piece of code that, looks for some surface linguistics construct and takes a specific action if it is found. It consists of a test condition, to be searched for in the input, and an action, to be executed if the test is successful. The test might be as specific as a particular word or as general as an entire conceptualization.

  The action might contain information about:

  1. What to look for next in the input;
  2. What to do with the input just found;
  3. How to organize the representation.

The flexibility of this formalism allows the system to function without depending heavily on syntax, although it is otherwise quite similar to the tests and actions that make ATN's (Augmented Transition Network) such a powerful parsing mechanism.

- Inference module: It was written by Charles Rieger. The inference module can accept a proposition (stated in CD representation) and deduce a large number of facts from the proposition in the current context of the system's memory. The reason behind this component was the assumption that humans understand far more from a sentence than is actually stated.

Sixteen types of references were identified, including cause, effect, specification, and function.

The inference knowledge was represented in memory in a modified semantic net, the inferences were organized into "molecules", for the purpose of applying them.

An example of this process might be:

John hit Mary

From which the system might infer (among many other things):

John was angry with Mary.
Mary might hit John back.
Mary might get hurt.

The module does relatively unrestricted forward inference process, which tended to produce large numbers of inferences for any given input.

- Text generation module: It was written by Neil Goldman. This took an internal CD representation and converted it into English-like output, in a two-part process:

  1. A discrimination net was used to distinguish between different word-senses. This permitted the system to use English-specific contextual criteria for selecting words (especially verbs) to "name" conceptual patterns.
  2. An ATN was used to linearize the CD representation into a surface-like structure.

MARGIE ran in two modes:

- Inference mode:
  MARGIE would accept a sentence and attempt to make inference from that sentence.
- Paraphrase mode:
  MARGIE would attempt to restate the sentence in as many equivalent ways as possible.

  For example, given the input

  John killed Mary by choking her.

  It might produce the paraphrases

  John strangled Mary.

  John choked Mary and she died because she was unable to breathe.

  Here, “strangle”: means “to kill by pressing on the throat”.

6.3.3 SAM (Script Applier Mechanism) and PAM (Plan Applier Mechanism)

SAM and PAM are computer programs developed by Roger Schank, Robert Abelson, and their students at Yale University to demonstrate the use of “scripts” and “plans” in understanding simple stories.

CD representation is the interlingua that is produced when SAM and PAM parses sentences. The parser for these programs is an extension of the one developed by Christopher Riesbeck for the MARGIE system. Further processing in SAM and PAM involves the manipulation of CD structures and of higher structures built on them – scripts, plans, goals and themes..

6.3.3.1 **Script**:: A script is a standardized sequence of events that describes some stereotypical human activity, such as going to a restaurant or visiting a medical doctor. Schank and Abelson’s assumption is that people know many such scripts and use them to establish the context of events. For example, the RESTAURANT script involves going to restaurant, being seated, consulting the menu, and so on. People who are presented with an abbreviated description of this activity, for example, the sentence “John went out to dinner”, infer from their own knowledge about restaurants that John ordered, ate, and paid for food. Moreover, they anticipate (expect) from a sentence that fills part of the script (e.g. “John was given a menu”) what sorts of sentences are likely to follow (e.g. “John ordered the lamb”)

Following is a script:

Players: customer, server, cashier

Props: restaurant, table, menu, food, check, payment, tip

Events:

1. customer goes to restaurant
2. customer goes to table
3. server brings menu
4. customer orders food
5. server brings food
6. customer eats food
7. server bring check

8. customer leaves tip for server
9. customer gives payment to cashier
10. customer leaves restaurant

Header: event 1

Main concept: event 6

Fig. 1

Two components of scripts are of special importance. The first important component is script header, in the restaurant script, it is event 1 (going to restaurant); the second important component is main concept, or goal of the script, in the restaurant script, the goal is to eat food.

The scripts in SAM grew out of Abelson (1973) notion of scripts as networks of causal connections. These connections depend on determining what actions result in what states, and what states enable what actions. There are four causal links that we use to connect together events and states:

1. Result causation: An event can have a result that changes some state of an object involved in the event. E.g. :"the lamp broke because John hit it"
2. Enable causation: When a state change occurs it may change the conditions of the world in such a way as to enable an event to take place that could not have taken place before. This is called enable causation. These state changes only enable potential events. That is, they do not directly cause the event to take place but only enable its potential occurrence. e.g. "John ate because there was food on the table".
3. Initiation causation: Whenever an event or state change occurs, or whenever a state or a potential event exists, it is possible that an actor may be made aware of it, and thus made to think about it. This is called initiation causation. Initiation causation accounts for people's thinking about things. E.g. "John realized that Mary was unhappy because he saw her crying".
4. Reason causation: Once people have started thinking about things, they are likely to decide to do something. (This is the primitive action MBUILD). Deciding to do something is the reason for doing it. Thus when we say that "John ate fish because he was hungry" we have an example of reason causation

. These causal connections are quite frequently in the texts. In fact, a great many texts could be seen, after an analysis that linked together all the causal connections, implied or explicit, to be one enormous causal chain of events and states.

Scripts describe everyday events, but frequently these events do not run to completion. For example,

I want to the restaurant.
I had a hamburger.
Then I bought some groceries.

This story presents several problems for a system like SAM that matches scripts to input sentences. One problem is that the restaurant script is "left dangling" by the introduction of the last sentence

It is not clear to the system whether the restaurant script

1. has terminated and a new (grocery shopping) script has started;
2. has been distracted by a "fleeting" (one sentence) grocery script; or

3. is interacting with a new grocery script (e.g. buying groceries in the restaurant)

Another thing that can happen to everyday scripts is that they can be thwarted, as in:

I went to the gas station to fill up my car.
But the owner said he was out of gas.

This is called an "obstacle".

Scripts describe rather special events, and although it is assumed that adults know thousands of them, story comprehension cannot be simply a matter of finding a script to match the story. There are just too many possible stories. Moreover, there are clear cases in which people comprehend a story even though it does not give enough information to cause a program to invoke a script, as in

John needed money.
He got a gun and went to a liquor store.

Schank and Abelson point out that even if the program had a script for "Robbery", this story offers no basis for invoking it. Nonetheless, people understand John's goals and his intended actions.
Schank and Abelson said:

"**There must be relevant knowledge available to tie together sentences that otherwise have no obvious connection… The problem is that there are a great many stories where the connection cannot be made by the techniques of causal chaining nor by reference to a script. Yet they are obviously connectable. Their connectability comes from these stories implicit reference to plans.** "

6.3.3.2 Plan:

Schank and Abelson introduce "plans" as the means by which goals are accomplished, and they state that understanding "plan-based" stories involved discerning the goals of the actor and the methods by which the actor chooses to fulfill those goals. The distinction between script-based and plan-based stories is very simple: In a script-based story, parts or all of the story correspond to one or more scripts available to the story understander; in a plan-based story, the understander must discern the goals of the main actor and the actions that accomplish those goals. An understander might process the same story by matching it with a script or scripts or by figuring out the plans that are represented in the story. The difference is that the script-based method is very specialized since a script refers to a specific sequence of actions, while plan-based method can be very general since the goals they accomplish are general. For example, in

John wanted to go to a movie. He walked to the bus stop.

We understand that John's immediate goal (called a "delta goal" or "D-goal" because it brings about a change necessary for accomplishment of the ultimate goal) is to get to the movie theatre. "Going somewhere" is a very general goal and does not apply just to going to the movies. This goal has associated with a set of "plan boxes", which are standard ways of accomplishing the goal. Plan boxes for "going somewhere" include riding an animal, taking public transportation, driving a car, and so on.
Obviously, a story understander might have a "Go to the movies" script in its repertoire, so that analysis of John's goals would be unnecessary – the system would just recognize the situation and

retrieve the script. This script can be thought of as the standardized intersection of a number of more or less general goals and their associated plan boxes. It would be a routinized plan made up of a set of general sub-plans; “Go to somewhere (the theater)”, “Purchase something (a ticket)”, “Purchase something (some popcorn)”, and so forth.

Schank and Abelson said: “**A routinized plan can become a script, at least from planner’s personal point of view. Thus, plans are where scripts come from. They compete for the same role in the understanding process, namely as explanations of sequences of actions that are intended to achieve a goal.**”

The process of understanding plan-based stories involves determining the actor’s goal, establishing the sub-goals (D-goals) that will lead to the main goal, and matching the actor’s actions with plan boxes associated with the D-goals.

For example, in

John was very thirsty. He hunted for a glass.

We recognize the D-goal of PTRANSing liquid and the lower level goal (specified in the plan box for PTRANSing liquid) of finding a container to do it with,

6.3.3.3 **Goals and Themes:**.

In story comprehension, goal and sub-goals may arise from a number of sources. For example, they may be stated explicitly, as in

John wanted to eat.

They may be nested in a plan box, or they may arise from “themes”.

For example, if a LOVE theme holds between John and Mary, it is reasonable to expect the implicit, mutual goal of protecting each other from harm:

Schank and Abelson said: “**themes, in other words, contain the background information upon which we base our predictions that an individual will have a certain goal**.”

Themes are rather like production systems in their situation-action nature. A theme specifies a set of actors, the situation they may be in, and the actions that will resolve the situation in a way consistent with the theme. The goals of a theme are to accomplish these actions. Schank and Abelson have proposed seven types of goals and some D-goals.

Other examples are:

A- or Achievement-goals:	To desire wealth is to have an A-Money goal.
P- or Preservation goal:	To protect someone may be a P-Health or P-Mental State goal.
C- or Crisis-goal:	A special case of P-goals, when action is immediately necessary.

The LOVE theme can be stated in terms of some of these goals:

X is the lover, Y is the loved one; Z is another person.

SITUATION	ACTION
Z cause Y harm	A-Health (Y) and possibly cause Z harm or C-Health (Y)
Not-love (Y, X)	A-Love (Y, X)

General goals: A-Respect (Y)
A-Marry (Y)
A-Approval (Y)

To summarize the knowledge structure we have discussed, we note their interrelationships as follows:

1. Themes give rise to goals.
2. A plan is understood when its goals are identified and its actions are consistent with the accomplishment of those goals.
3. Scripts are standardized models of events.
4. Scripts are specific; plans are general.
5. Plans originate from scripts.
6. Plans are ways of representing a person's goals. These goals are implicit in scripts, which represent only the actions.
7. A script has a header, which is pattern-matched to an input sentence. Plans do not have headers, but each plan is subsumed under a goal.

SAM and PAM are designed just based on these conceptions - scripts, plans, goals and themes.

Both SAM and PAM accept stories as input, both use an English-to-CD parser to produce an internal representation of the story (in conceptual dependency). Both can paraphrase the story and make intelligent inferences from it. They differ with respect to the processing that goes on after the CD representation has been built.

6.3.3.4 SAM system

SAM understands stories by fitting them into one or more scripts. After this match is completed, it makes summaries of the stories. The process of fitting a story into a script has three parts:

■ PARSER:

The PARSER generates a CD-representation of each input sentence. But the PARSER does not make many inferences; thus, it does not realize that "it" refers to the "hot dog" in

The hot dog was burned. It tasted awful.

This task is left to MEMTOK.

■ MEMTOK

The MEMTOK ia a memory module. This module takes rrferences to people, places, things, and so forth and fills in information about yjem.It recognizes that the "it" in the sentence above refers to the "hot dog", and instantiates the "it" node in the CD representation of the second sentence with the "hot dog" node from the first sentence.

■ APPLY

The APPLY is the script applier. This module has three functions:

① It takes a sentence from PARSER and checks whether it matches the current script, a concurrent script, or any script in the database. If this matching is successful, then it carries out second function.

② It makes a set of predications about likely inputs to follow.

③ It instantiates any steps in the current script that were skipped over in

the story.

When the whole story has been mapped into a CD representation, the SAM program can produce a summary of the story or answer question about it. SAM can produce summaries in English, Chinese, Russian, Dutch, and Spanish.

Following is an example of paraphrase by SAM system.

ORIGINAL John went to a restaurant. He sat down. He got mad. He left.

PARAPHRASE: JOHN WAS HUNGRY. HE DECIDED TO GO TO A RESTAURANT. HE WENT TO ONE. HE SAT DOWN IN A CHAIR. A WAITER DID NOT GO TO THE TABLE. JOHN BECAME UPSET. HE DECIDED HE WAS GOING TO LEAVE THE RESTAURANT. HE LEFT IT.

SAM inferred that John left restaurant because he did not get any service. The basis for this reference is that in the restaurant script, event 3 represents the waiter coming over to the table after the main actor has been seated. SAM knows that people can get mad if their expectation are not fulfilled and infers that John's anger results from the nonoccurrence of event 3.

6.3.3.5 PAM system

PAM program was written by Wilensky (1978). The process on understanding paln-based stories in PAM is as follows:

- Determine the goal;
- Determine the D-goals that will satisfy that goal;
- Analyze input concepualization for their potential realization of one of the plan-boxes that are called by one of the determined D-goals.

PAM utilizes two kinds of knowledge structures in understanding goals: named plans and themes. A named plan is a set of actions and sub-goals for accomplishing a main goal. It is not very difficult from a script. When PAM encounters a goal in a story for which it has a named plan, it can make predictions about the D-goals and the actions that will follow. It will look for these D-goals and actions on subsequent inputs. Finding them is equivalent to understanding the story.

Themes provide another source of goals for PAM.

Consider the sentences:

1. John wanted to rescue Mary from the dragoon.
2. John loves Mary. Mary was stolen away by a dragoon.

In both of these cases, PAM will expect John to take action that are consistent with the goal of rescuing Mary from dragoon, even though this goal was not explicitly mentioned in sentence 2. The source of this goal on sentence 2 is, the LOVE theme mentioned above – if another actor tries to cause harm to a loved one, the main actor sets up the goal of Achieving-Health of the loved one and possible harming the evil part..(It is assumed that the dragoon stole Mary in order to hurt her).

PAM determines the goals of an actor by

- Noting their explicit mention in the text of the story;
- Establishing them as D-goals for some known goal;
- Inferring them from a theme mentioned in the story

In PAM, to understand a story is to "**keep track of the goals of each of the characters in a story and to interpret their actions as means of achieving those goals.**" (Schank and Abelson).

The program begins with written English text, converts it into CD representation, and then

interprets each sentence in term of goals or actions themselves. When this process is completed, PAM can summarize the story and answer questions about the goals and actions of the characters.

6.3.4 TALE-SPIN

TALE-SPIN (Lames Meehan, 1976) is a program that writes simple stories. The writing of TALE-SPIN embodied the traditional AI cycle of research.

- Step 1: to define the story;
- Step 2: to write a program modeling the CD theory and to add it to the existing system;
- Step 3: to run the system and to observe where the model was incorrect or inadequate, thereby identifying the need for some more theory.

Following is an example of TALE-SPIN about the “thirsty”.
After an initial dialogue with the user, establishing the characters and the physical setting, the story begins when each of the characters is made thirsty.

“Once upon a time George ant lived near a patch of ground. There was a nest in an ash tree. Wilma bird lived in the nest. There was some water in a river. Wilma knew that the water was in the river. George knew that the water was in the river. One day Wilma was very thirsty. Wilma wanted to get near some water. Wilma flew from her nest across a meadow through a valley to the river. Wilma drank the water. Wilma was not thirsty any more.

George was very thirsty. George wanted to get near some water. George walked form his patch of ground across the meadow through the valley to a river bank. George fell into the river. George wanted to get near the valley. George couldn’t get near the valley. George wanted to get near the meadow. Wilma wanted George to get near the meadow. Wilma wanted to get near George. Wilma grabbed George with her claw. Wilma took George from the river through the valley to the meadow. George was devoted to Wilma. George owed everything to Wilma. Wilma let go of George. George fell to the meadow. The end.”

6.3.5 POLITICS

The POLITICS system (Carbonell, 1979) is a set of closely cooperating computer programs that simulate human ideological understanding of international political events. The system simulates a conversation with a United State Senator on a politically sensitive issue. POLITICS adopts a political perspective in the conversation, either conservative or liberal, and evaluate the import of any news event on the basis of this viewpoint. The program has its own belief system, and a series of goals trees which rank political goals (or beliefs) into a hierarchy of relative importance.

The representation of political ideologies must contain all the information necessary for the reasoning mechanism to simulate widely differing ideological interpretation. The simulation system should be able to model different ideological behavior.

Following is the conversation by POLITICS on the event “The US congress is expected to approve the Panama Canal Treaty”.

Conservative viewpoint:

Q1: should the US approve the treaty?
A1: NO, THE TREATY IS BAD FOR THE UNITED STATES.

Q2: Why is the treaty bad for the US?
A2: THE UNITED STATES WOULD LOSE THE CANAL TO PANAMA AND THE UNITED STATES WILL BE WEAKER.

Q3: What might happen if the United States loses the Canal?
A3: RUSSIA WILL TRY TO CONTROL THE CANAL.

Q4: Why would Russia try to take over the canal?
A$: RUSSIA WANTS TO EXPEND ITS MILITARY CONTROL. THE PANAMA CANAL HAS HIGH MILITARY VALUE.

Q5: How might this happen?
A5: THE UNITED STATES MAY NOT BE ABLE TO STOP ANYONE TAKING OVER THE PANAMA CANAL. TORRIJOS IS A COMMUNIST. TORRIJOS WILL LET RUSSIA CONTROL PANAMA.

Q6: Why should the US worry about Russia taking over the canal?
Q6: BECAUSE RUSSIA WANTS TO CONTROL THE WORLD>

Q7: What should the US do to stop Russia from taking over the Canal”
A7: CONGRESS SHOULD VOTE AGAINST THE PANAMA CANAL TREATY.

Liberal viewpoint:

Q1: Should the US approve the treaty?
A1: YES. THE PANAMA CANAL TREATY IS GOOD FOR THE UNITED STATES.

Q2: Why is the treaty good?
A2: THE PANAMA CANAL TREATY IMPROVES RELATIONS WITH LATIN AMERICAN COUNTRIES.

Q3: What happens if the US keeps the canal?
A3: THIS WILL CAUSE DISCONTENT AND CONFLICT IN PANAMA AND LATIN AMERICA.

Q4: Why would this lead to conflict?
A4: THE CANAL ZONE IS A UNITED STATES COLONY. LATIN AMERICA DOES NOT LIKE COLONIALISM. THE UNITED STATES WOULD MAINTAIN COLONIALISM.

Q5: What if Russia tries to take over the canal?
A5: RUSSIA DOES NOT WANT CONFLICT WITH THE US. RUSSIA WILL STAY OUT OF

THE CANAL, IF THE UNITED STATES DEFENDS IT. THE UNITES STATES DEFENDING THE CANAL IS PART OF THE TREATY.

The Panama Canal event is the most complicated type of scenario that the program can understand. POLIYICS is limited in this domain of application to interpreting the political events, but it may use its mechanisms in such as counter-planning process that are much more generally applicable.