Chapter 5

Higher-Order Inference

The NAL built so far is “first-order,” in the sense that statements are relations among terms, but a statement cannot be treated as a term. In “higher-order inference,” a statement can be treated as a term, therefore there are *statements on statements*, as well as inference on this kind of higher-order statements. In NAL, though it is possible to further divide higher-order statements into second-order, third-order, and so on, such a distinction is not practically useful. Therefore, they will be covered together under the same notion of “higher-order statements.” In this chapter, NAL will be extended, step by step, to include various types of higher-order statements, as well as the inference on them.

5.1 NAL-5: statements as terms

In NAL-5, new grammar and inference rules are introduced, so that the system can treat a statement as a term.

5.1.1 Higher-order statements

First, the new grammar rules of Narsese-5 are listed in Table 5.1. In NAL-5 a statement can be used as a term (so it can be involved in various inheritance and ordinary relations). Some ordinary relations, such as “believe,” “say,” “know,” and so on, take a statement as an argument. For example, “John knows that whale is a kind of mammal” is

$<term>$	$::=$	$(<statement>)$
$<statement>$	$::=$	$<term>$
		$\mid (\neg <statement>)$
		$\mid (\wedge <statement><statement>^+)$
		$\mid (\vee <statement><statement>^+)$
$<copula>$	$::=$	$\Rightarrow \mid \Leftrightarrow$

Table 5.1: The New Grammar Rules of Narsese-5

represented in Narsese as "$(\{John\} \times \{whale \rightarrow mammal\}) \rightarrow know$," where the subject term is a product containing a statement.

On the other hand, a term can also be used as a statement (that corresponds to the name of a statement, such as "Newton's first law"). However, it does not mean that there is no difference between *term* and *statement*. In NAL, a statement has both meaning and truth value, but a non-statement term only has meaning (without truth value).

Compound statements can be formed using statement operators *negation* ("$\neg$"), *conjunction* ("$\wedge$"), and *disjunction* ("$\vee$"), whose meanings are intuitively similar to those in propositional logic.[1] As usual, conjunction and disjunction can be used both in the prefix format and the infix format.

Finally, two new copulas, *implication* ("$\Rightarrow$") and *equivalence* ("$\Leftrightarrow$"), are defined between statements. Intuitively, they correspond to "if" and "if-and-only-if," respectively. They are "higher-order relations" because they are only defined between two statements.

Please note that in general "$\Rightarrow$" and "$\Leftrightarrow$" are different from "$\supset$" and "$\equiv$," though their intuitive meanings are similar. The former two belong to the object language (Narsese), while the latter two belong to the meta-language of Narsese (propositional calculus). The two new copulas in NAL have different status from the three statement operators

[1]In this book, though the same symbols are used for the statement operators in Narsese and the logical connectives in the meta-language of NAL (as defined in propositional calculus), they should be distinguishable by context.

mentioned above, whereas in propositional calculus the five corresponding notions have the same status as truth-value operators.[2]

5.1.2 Implication and inheritance

First, the implication relation is defined by valid inference in NAL.

Definition 34 *If S_1 and S_2 are statements, "$S_1 \Rightarrow S_2$" is true (i.e., has truth value $<1, 1>$) if and only if from "$S_1 <1, 1>$" alone NAL can derive "$S_2 <1, 1>$."*

The derivation in the above definition can consists of any (finite) number (0, 1, 2, ..., n) of inference steps.

Theorem 34 *The implication relation, "$\Rightarrow$," is a reflexive and transitive relation from one statement to another statement.*

Since the above theorem of implication is parallel to the definition of inheritance (in NAL-0), higher-order inference in NAL can be defined as partially isomorphic to first-order inference. The correspondences are listed in Table 5.2.

first-order inference	**higher-order inference**
inheritance	implication
similarity	equivalence
subject	antecedent
predicate	consequent
extension	sufficient condition
intension	necessary condition
extensional intersection	conjunction
intensional intersection	disjunction

Table 5.2: The Isomorphism Between First-Order and Higher-Order

The definitions of the new notions in Table 5.2 are in the following.

Definition 35 *An* implication statement *consists of two statements related by the implication relation. In implication statement "$A \Rightarrow C$," A is the* antecedent*, and C is the* consequent*.*

[2]This difference will be discussed in detail in subsection 9.4.1.

Definition 36 *Given experience K, the* sufficient conditions *of a statement T is the set of statements $T^S = \{x \mid x \in V_K \wedge x \Rightarrow T\}$; the* necessary conditions *of T is the set of statements $T^N = \{x \mid x \in V_K \wedge T \Rightarrow x\}$.*

Definition 37 *For an implication statement "$A \Rightarrow C$," its* evidence *are statements in A^S and C^N. Among them, statements in $(A^S \cap C^S)$ and $(C^N \cap A^N)$ are* positive evidence, *while statements in $(A^S - C^S)$ and $(C^N - A^N)$ are* negative evidence.

Definition 38 *The* equivalence *relation is a symmetric implication relation. That is, "$A \Leftrightarrow B$" is defined to mean "$(A \Rightarrow B) \wedge (B \Rightarrow A)$."*

The amounts of evidence and the truth value for a higher-order statement are defined in the same way from evidence as for first-order statements.

Now the meaning of a statement includes not only its extension and intension, but also its sufficient and necessary conditions.

Please note that the truth value of an implication (or equivalent) statement do not depend on all its inheritance (or similarity) relations with other terms. Two statements can be fully equivalent (i.e., "$P \Leftrightarrow Q$" has a frequency $= 1$), but still have different meanings (i.e., "$P \leftrightarrow Q$" has a frequency < 1). On the other hand, if two statements have the same meaning, they should also have the same truth value.

Definition 39 *When S_1 and S_2 are different statements, their* conjunction, *$(S_1 \wedge S_2)$, is a compound statement defined by*

$$(\forall x)((x \Rightarrow (S_1 \wedge S_2)) \equiv ((x \Rightarrow S_1) \wedge (x \Rightarrow S_2))).$$

Their disjunction, *$(S_1 \vee S_2)$, is a compound statement defined by*

$$(\forall x)(((S_1 \vee S_2) \Rightarrow x) \equiv ((S_1 \Rightarrow x) \wedge (S_2 \Rightarrow x))).$$

The above two statement operators are symmetric, and can be extended to take more than two arguments.

Because of this isomorphism, there is an isomorphic inference rule in NAL-5 for the following rules defined previously:

- The NAL-1 rules for choice, revision, deduction, abduction, induction, exemplification, and conversion.
- The NAL-2 rules for revision, comparison, analogy, conversion, and deduction.
- The NAL-3 rules for the processing of intersections.

The term operators for (extensional/intensional) set, product, and (extensional/intensional) image are not involved in the isomorphism between first-order and higher-order terms. They treat higher-order terms just like first-order terms, and there is no special rule added.

To treat conditional statements (implications) and categorical statements (inheritances) in a similar way is not a new idea. The following opinion can be traced back to Leibniz: "Conditionals are categorical by virtue of the fact that the relationship between an antecedent and a consequent is exactly that the relationship between a subject and a predicate, namely, containment" [Englebretsen, 1981]. In predicate calculus, categorical statements are translated into conditional statements. What makes NAL different is that it does not treat the two as *the same*, but as *isomorphic to each other*. Consequently, the corresponding inference rules have different forms and meanings, though using the same truth-value function.

5.1.3 Implication as conditional statement

Besides the above isomorphism, higher-order inference and first-order inference in NAL can be directly related to each other, by extending the identity between an implication statement ($S_1 \Rightarrow S_2$) and an inference process (from S_1 to S_2) to actual judgments.

By definition, in NAL a judgment "$S < f,\ c >$" indicates that "The degree of belief the system has on statement S, according to available evidence, is measured by truth value $< f,\ c >$." Now if we assume that the available evidence currently used on the evaluation of S can be written as a compound statement E, then the same meaning can be represented by "$E \Rightarrow S < f,\ c >$," that is, "The degree of belief the system has on statement 'If E is true, then S is true' is measured by truth value $< f,\ c >$." In this way, a statement S is equivalently transformed into an implication statement "$E \Rightarrow S$" ("If the available evidence is true, then S is true"). This transformation is justified according to the semantics of NAL.

This transformation is a conceptual one, not an actual one in the sense that there is a statement used by NAL corresponding to the above E. This conceptual transformation is used to justify inference rules. We can add this implicit conditions into the premises, so as to change the premise combinations into the ones for which we already have inference rules. Finally, we remove the implicit condition from the conclusion. Table 5.3 contains several rules obtained in this way.

premises	**add condition**	**conclusion**	**drop condition**
$M \Rightarrow P,\ M$	$M \Rightarrow P,\ E \Rightarrow M$	$E \Rightarrow P$	$P\ <F_{ded}>$
$P \Rightarrow M,\ M$	$P \Rightarrow M,\ E \Rightarrow M$	$E \Rightarrow P$	$P\ <F_{abd}>$
$M \Leftrightarrow P,\ M$	$M \Leftrightarrow P,\ E \Rightarrow M$	$E \Rightarrow P$	$P\ <F'_{ana}>$

Table 5.3: The Conditional Syllogistic Rules (1)

Similarly, when the two premises can be seen as derived from the same evidence, it can be used as the common virtual condition of the two, and some conclusions can be derived accordingly, as in Table 5.4.

premises	**add condition**	**conclusion**	**drop condition**
$P,\ M$	$E \Rightarrow P,\ E \Rightarrow M$	$M \Rightarrow P$	$M \Rightarrow P\ <F_{ind}>$
$P,\ M$	$E \Rightarrow P,\ E \Rightarrow M$	$M \Leftrightarrow P$	$M \Leftrightarrow P\ <F_{com}>$
$P,\ M$	$E \Rightarrow P,\ E \Rightarrow M$	$E \Rightarrow (P \wedge M)$	$P \wedge M\ <F_{int}>$
$P,\ M$	$E \Rightarrow P,\ E \Rightarrow M$	$E \Rightarrow (P \vee M)$	$P \vee M\ <F_{uni}>$

Table 5.4: The Composition Rules of NAL-5

The truth-value functions in Table 5.3 and Table 5.4. are those defined in NAL-1 to NAL-3. For practical purpose, we can ignore the two columns in the middle, and treat the rules as directly go from the first column (premises) to the last column (conclusions).

Now we have three groups of syllogistic rules (deduction, abduction, and induction), one defined on the inheritance relation (in NAL-1), one on the implication relation (in the previous subsection), and one on a mixture of the two (above). In the three groups, each type of inference (deduction, abduction, or induction) has a different form, but uses the same truth-value function. The last group is similar to how the three types of inference are defined in extended propositional calculus [Flach and Kakas, 2000], except that in NAL the statements have truth values attached to indicate their evidential support.

Also according to the semantical interpretation of implication, we have

$$(M \Rightarrow ((\wedge\, A_1 \cdots A_m) \Rightarrow C)) \equiv ((\wedge\, M A_1 \cdots A_m) \Rightarrow C)$$

that is, a conditional statement of a conditional statement is equivalent to a conditional statement with a conjuncted condition. This is similar to what is called "exportation" in propositional logic [Copi, 1982]. This equivalence give us the rules in Table 5.5.

$\{(\wedge\, M A_1 \cdots A_m) \Rightarrow C,\ M\} \vdash (\wedge\, A_1 \cdots A_m) \Rightarrow C$	$<F_{ded}>$
$\{(\wedge\, M A_1 \cdots A_m) \Rightarrow C,\ (\wedge\, A_1 \cdots A_m) \Rightarrow C\} \vdash M$	$<F_{abd}>$
$\{(\wedge\, A_1 \cdots A_m) \Rightarrow C,\ M\} \vdash (\wedge\, M A_1 \cdots A_m) \Rightarrow C$	$<F_{ind}>$

Table 5.5: The Conditional Syllogistic Rules (2)

The truth values of the premises are omitted in the rules. As before, the induction rule is applied only when the two premises are based on the same evidence.

These rules can be seen as generalizations of the previous table where $m = 0$.[3] Table 5.6 gives further extension of these rules.

In each group of the syllogistic rules, abduction and induction can be obtained from deduction by switching a (different) premise and the conclusion, so they are "reversed deduction" in a sense.

With the help of the isomorphism and the implicit condition technique, we also get the following implications in NAL-5 among statements.

[3]So C is $(\wedge\, A_1 \cdots A_m) \Rightarrow C$, and $M \Rightarrow C$ is $(\wedge\, M A_1 \cdots A_m) \Rightarrow C$.

$\{(\wedge\ MA_1 \cdots A_m) \Rightarrow C,\ A_0 \Rightarrow M\} \vdash (\wedge\ A_0A_1 \cdots A_m) \Rightarrow C$	$<F_{ded}>$
$\{(\wedge\ MA_1 \cdots A_m) \Rightarrow C,\ (\wedge\ A_0A_1 \cdots A_m) \Rightarrow C\} \vdash A_0 \Rightarrow M$	$<F_{abd}>$
$\{(\wedge\ A_0A_1 \cdots A_m) \Rightarrow C,\ A_0 \Rightarrow M\} \vdash (\wedge\ MA_1 \cdots A_m) \Rightarrow C$	$<F_{ind}>$

Table 5.6: The Conditional Syllogistic Rules (3)

Theorem 35

$$\begin{aligned} (S_1 \wedge S_2) &\supset S_1 \\ S_1 &\supset (S_1 \vee S_2) \end{aligned}$$

Now we can also turn the (meta-level) implications in NAL theorems into inference rules. Since a meta-level implication theorem "$A \supset C$" corresponds to inference from statement A to statement C, it can be treated as an implication statement in Narsese "$A \Rightarrow C < 1, 1 >$," which can be used with "$A < f, c >$" to get "$C < f, fc >$," using the deduction rule. Similarly, each meta-level implication theorem "$A \supset C$" can also be used with "$C < f, c >$" to get "$A < f, c/(c + k) >$" as a special kind of abduction. Isomorphically, theorems in the form of inheritance statements can also be used in this way.

The above inference has some restrictions. For example, "$S_1 \Rightarrow (S_1 \vee S_2) < 1, 1 >$" is a theorem, and it is fine to use it and a judgment of S_1 to derives a judgment of $(S_1 \vee S_2)$ by deduction, where S_2 can be any term. However, this result should not be used with the theorem again to derive a judgment of S_2 by abduction, otherwise for any two arbitrary statements, from the truth value of one, the truth value of the other can be derived (though with a low confidence).

The equivalence statements in theorems and definitions are easier to be converted into inference rules: "$S_1 \equiv S_2$" corresponds to rules $\{S_1\} \vdash S_2$ and $\{S_2\} \vdash S_1$, where the truth value of the conclusion is the same as the premise. The same result can be obtained by treating the inference as a special case of analogy with "$S_1 \Leftrightarrow S_2 < 1, 1 >$." Isomorphically, theorems in the form of similarity statements can also be used in this way.

5.1.4 Negation

Since the negation operator in NAL-5 takes one argument, it is not directly isomorphic to the (extensional/intensional) difference operators defined in NAL-3. Instead, it is defined as the following:

Definition 40 *If S is a statement, its* negation*, $(\neg S)$, is a compound statement defined by switching the positive and negative evidence of S.*

The definition also gives us the negation rule, defined in Table 5.7.

$$\boxed{\{S < f_0, c_0 >\} \vdash (\neg S) < F_{neg} >}$$

Table 5.7: The Negation Rule

The negation rule use the following truth-value function:

$$F_{neg}: \ f = 1 - f_0, \ c = c_0$$

We still have the following theorems as in propositional logic:

Theorem 36 $(\neg(\neg S)) \equiv S$

Theorem 37 $(S_1 \Leftrightarrow S_2) \equiv ((\neg S_1) \Leftrightarrow (\neg S_2))$

However, the *Law of Contrapositive* in propositional logic [Copi, 1982] (i.e., the equivalence between "$S_1 \Rightarrow S_2$" and "$(\neg S_2) \Rightarrow (\neg S_1)$") is no longer true in NAL.

When the truth values of S_1 and S_2 are determined according to certain evidence E, the induction rule can be used to calculate the truth value of "$S_1 \Rightarrow S_2$." As a result, when both S_1 and S_2 are true, it is positive evidence for "$S_1 \Rightarrow S_2$"; when S_1 is true but S_2 is false, it is negative evidence (when S_1 is false, it is not evidence).

Similarly, when both S_1 and S_2 are false, it is positive evidence for "$(\neg S_2) \Rightarrow (\neg S_1)$" (because both $\neg S_1$ and $\neg S_2$ are true); when S_2 is false but S_1 is true, it is negative evidence (when S_2 is true, it is not evidence).

By comparing the above two cases, we can see that "$S_1 \Rightarrow S_2$" and "$(\neg S_2) \Rightarrow (\neg S_1)$" have the same negative evidence (S_1 true, S_2 false), but completely different positive evidence ("both true" for the former, and "both false" for the latter). Therefore, to derive one statement from the other, in NAL we use the *contraposition rule* defined in Table 5.8.

$$\boxed{\{S_1 \Rightarrow S_2 < f_0, c_0 >\} \vdash (\neg S_2) \Rightarrow (\neg S_1) < F_{cnt} >}$$

Table 5.8: The Contraposition Rules of NAL-5

Since only negative evidence is passed from the premise to the conclusion, and the premise at most is counted as evidence with unit amount, we have $w = (1 - f_0)c_0$, and therefore the truth-value function is:

$$F_{cnt}: \; f = 0, \; c = (1 - f_0)c_0/[(1 - f_0)c_0 + k]$$

To summarize, related to the traditional study of "conversion," "obversion," and "contraposition" [Copi, 1982], in NAL the corresponding relations with evidence are in Table 5.9.

$S_1 \Rightarrow S_2$ $\{S_1 \wedge S_2; S_1 \wedge (\neg S_2)\}$	$S_2 \Rightarrow S_1$ $\{S_1 \wedge S_2; (\neg S_1) \wedge S_2\}$
$(\neg S_1) \Rightarrow (\neg S_2)$ $\{(\neg S_1) \wedge (\neg S_2); (\neg S_1) \wedge S_2\}$	$(\neg S_2) \Rightarrow (\neg S_1)$ $\{(\neg S_1) \wedge (\neg S_2); S_1 \wedge (\neg S_2)\}$

Table 5.9: Negation and Evidence

In Table 5.9, there are four statements, each with its positive evidence and negative evidence listed under it. There are three relations among them:

conversion. The two statements in the same row become each other by exchanging the subject and the predicate. They have the same positive evidence, but different negative evidence.

obversion. The two statements in the same column become each other by negating the subject and the predicate. They have different positive and negative evidence.

contraposition. The two statements in the same diagonal line become each other by exchanging and negating the subject and the predicate. They have the same negative evidence, but different positive evidence.

Given the above relations, in NAL there are rules for conversion and contraposition, but not for obversion, since no evidence can be passed from the premise to the conclusion. For the former two, only one type of evidence is passed (positive in conversion and negative in contraposition), so the frequency value of the conclusion is a constant (1 in conversion and 0 in contraposition). In these two rules, at most a single unit of evidence can be provided, so the conclusion has a lower confidence value than the premise — this is the case, because in NAL, the evidence of a premise is never directly taken as evidence (of the same quality and quantity) of the conclusion, except in the revision rule, where the premises and the conclusion are all about the same statement.

By definition, the evidence of $(\neg(S_1 \Rightarrow S_2))$ is obtained by switching the positive and negative evidence of $(S_1 \Rightarrow S_2)$, so it is $\{S_1 \wedge (\neg S_2); S_1 \wedge S_2\}$. Since it is nothing but the evidence of $(S_1 \Rightarrow (\neg S_2))$, the two statements are equivalent. This relation is not true if "$\Rightarrow$" is interpreted as the material implication in propositional logic. It is also important to notice that in NAL, unlike in propositional calculus, "$A \Rightarrow C$" is usually different from "$(\neg A) \vee C$" in truth value.[4] Similarly, for the equivalence relation, there are two other equivalence rule. These equivalence rules are listed in Table 5.10.

Other negation-related inference rules include the ones obtained from the following implications (which are also true in propositional logic):

[4]This issue is discussed in detail in subsection 9.4.1.

$\neg(S_1 \Rightarrow S_2)$	$S_1 \Rightarrow (\neg S_2)$
$\neg(S_1 \Leftrightarrow S_2)$	$S_1 \Leftrightarrow (\neg S_2)$
$\neg(S_1 \Leftrightarrow S_2)$	$(\neg S_1) \Leftrightarrow S_2$

Table 5.10: The Equivalence Rules of NAL-5

Theorem 38

$$\begin{array}{rcl} (\neg A) & \supset & (\neg(A \wedge B)) \\ (\neg(A \vee B)) & \supset & (\neg A) \\ (A \wedge (\neg(A \wedge B))) & \supset & (\neg B) \\ ((\neg A) \wedge (A \vee B)) & \supset & B \end{array}$$

So far, we have seen three types of (statement level) negation. For a statement "$S \rightarrow P$" in NAL-0, its negation is implicitly represented, which is not a statement in Narsese-0, though we can talk about "$\neg(S \rightarrow P)$" in its meta-language, where "$\neg$" is used as in propositional logic. In NAL-5, a statement S is multi-valued (with truth value $<f,\, c>$), and so is $\neg S$ (with truth value $<1 - f,\, c>$), where "$\neg$" is a statement operator defined in Narsese-5, whose meaning is not exactly the same as the one in predicate logic.

Finally, as described previously, "$S \rightarrow P$" in NAL-0 becomes "$S \rightarrow P < 1,\, 1 >$" in NAL-1 (as well as in all of its extensions, including NAL-5). However, when it is not true, its truth value $<f,\, c>$ can be anything except $< 1,\, 1 >$, and it is not necessarily the case specified by the negation of the statement, which has truth value $<0,\, 1>$. This issue is very important when encoding knowledge in Narsese. Usually, by "S is not P," we mean that "There is negative evidence for $S \rightarrow P$" (i.e., "$S \rightarrow P <0,\, c>$," where $c < 1$), but not that "All evidence for $S \rightarrow P$ is negative" (i.e., "$S \rightarrow P <0,\, 1>$").

5.2 NAL-6: statements with variables

5.2.1 Variable terms

The terms we introduced so far are *constant* terms, in the sense that at any given time, each of them is unique in the system, and has a determined meaning; a *variable* term, on the other hand, may appears in more than one statements in the system, each of them with its own meaning.

In NARS, the meaning of a constant term is determined by its experienced relations with the other term *in the whole system*, while the meaning of a variable term is locally determined by its relations with other terms *within the same statement*. That means, if there are two variable terms with the same name but not in the same statement, they are not necessarily related to each other in meaning. On the contrary, occurrences of a constant term in different statements are always bounded together.[5]

There are two types of variable terms in NAL: *independent variables* (similar to the *universal variables* in first-order predicate logic) and *dependent variables* (similar to the *existential variables* and *Skolem functions* in first-order predicate logic). The latter is a function of the former. A dependent variable has a (maybe empty) independent variable list in it to indicate its dependence. In Narsese, a variable term (of either the above two types) is named by a word (or a number) preceded by '#'.

Given AIKR, no statement in Narsese is made about all terms. Whenever a statement is made about a group of terms, it is usually possible to put them into the extension or intension of a given term, then to make a statement about it. A dependent variable indicates a single term under the given condition; an independent variable indicates any term under the given condition.

[5]Since in NARS the meaning of a constant term may change over time, the distinction between constant terms and variable terms is not whether the meaning of a term changes or not. In the next chapter, we will see that a constant term in Narsese is the name of a specific *concept* in NARS, while a variable term indicates an unspecified concept.

In Narsese, variable terms can only appear in special positions. An independent variable always appears in both sides of an implication or equivalence relation, as extension or intension of two terms. A dependent variable always appear in two components of a conjunction, also as extension or intension of two terms. Therefore, the following are the simplest statements with variable terms.

$$\begin{array}{ll} (\#x \to S) \Rightarrow (\#x \to P) & (\#x() \to S) \wedge (\#x() \to P) \\ (S \to \#x) \Rightarrow (P \to \#x) & (S \to \#x()) \wedge (P \to \#x()) \end{array}$$

In this way, the scope of a variable is the statement in which it appears. The name of a variable is arbitrary, as far as it is unique in the statement. In statements with multiple variables, each of them uses a different name, therefore its scope does not need to be explicitly specified — it is the smallest statement that contains all occurrences of the variable.

As in predicate logic, the scope of a variable can be embedded in that of another one. For example, in Narsese the following situations can be represented:

- $((\#x \to key) \wedge (\#y \to lock)) \Rightarrow ((\#x \times \#y) \to open)$
  ["Every key can open every lock."]
- $(\#x() \to key) \wedge ((\#y \to lock) \Rightarrow ((\#x() \times \#y) \to open))$
  ["There is a key that can open every lock."]
- $(\#x \to key) \Rightarrow ((\#y(\#x) \to lock) \wedge ((\#x \times \#y(\#x)) \to open))$
  ["Every key can open a lock."]
- $(\#x() \to key) \wedge (\#y() \to lock) \wedge ((\#x() \times \#y()) \to open)$
  ["There is a key that can open a lock."]

Since an inheritance relation is identical to two subclass relations of the extension and intension of the two terms, independent variables becomes implicit in first-order NAL. Using the variable terms introduced above, we can see that "$S \to P$" is equivalent to "$((\#x \to S) \Rightarrow (\#x \to P)) \wedge ((P \to \#y) \Rightarrow (S \to \#y))$." It roughly means "For any term x in the extension of term S, it is also in the extension of term P; for any term y in the intension of term P, it is also in the intension of term S."

Variable terms become necessary when the extension or intension of a term needs to be specified separately, as well as when complicated relations among terms need to be described. For example, "$(\#x \rightarrow S) \Rightarrow (\#x \rightarrow P)$" and "$(P \rightarrow \#y) \Rightarrow (S \rightarrow \#y)$" have different evidence, and can be separately maintained to represent pure extensional or intensional relations between terms.

The independent variables in NAL are different from the universal variables in first-order predicate logic in that the former are restricted by their relations with other terms in the statement. For example, in "$(\#x \rightarrow S) \Rightarrow (\#x \rightarrow P)$," the independent variable '$\#x$' only represents terms in the extensions of S, but not the ones that are not there. On the contrary, in a predicate logic proposition "$(\forall x)(S(x) \supset P(x))$," the universal variable x can represent every individual in the domain. Roughly speaking, the former says "Every instance of S is also an instance of P," while the latter says "For everything in the domain, if it is an instance of S, then it is also an instance of P." The meaning of the two sentences are not exactly the same, though related.[6]

5.2.2 Inference with variable terms

Since an independent variable represents a unspecified term (under a certain condition), it can be replaced by a specific term (satisfying the condition) without changing the truth value of a judgment. Technically speaking, an independent variable can be substituted by another (variable or constant) term that has the same relation by *unification* (as defined in predicate logic), and then a conclusion can be derived. Some variable elimination rules are listed in Table 5.11.

$\{(\#x \rightarrow S) \Rightarrow (\#x \rightarrow P),\ M \rightarrow S\}$	$\vdash$	$M \rightarrow P$	$<F_{ded}>$
$\{(\#x \rightarrow S) \Rightarrow (\#x \rightarrow P),\ M \rightarrow P\}$	$\vdash$	$M \rightarrow S$	$<F_{abd}>$
$\{(\#x \rightarrow S) \Leftrightarrow (\#x \rightarrow P),\ M \rightarrow S\}$	$\vdash$	$M \rightarrow P$	$<F'_{ana}>$

Table 5.11: Sample Independent-Variable Elimination Rules

[6] I will say much more about the difference between NAL and first-order predicate logic in Chapter 9 and 10.

Such a rule can be seen as a substitution (with $\#x$ replaced by M) followed by an inference defined in Table 5.3. The same technique can be applied to the other higher-order inference rules to use premises with variables.

The reverse of variable elimination rules introduces independent variables into conclusions, as listed in Table 5.12. These rules are justified in the same way as the rules in NAL-1 and NAL-2, except that here the "extensional inheritance" and "intensional inheritance" between S and P are separated, due to the using of a variable term.

$\{M \rightarrow P,\ M \rightarrow S\}$	$\vdash$	$(\#x \rightarrow S) \Rightarrow (\#x \rightarrow P)$	$<F_{ind}>$
$\{M \rightarrow P,\ M \rightarrow S\}$	$\vdash$	$(\#x \rightarrow S) \Leftrightarrow (\#x \rightarrow P)$	$<F_{com}>$

Table 5.12: Sample Independent-Variable Introduction Rules

The rule in Table 5.13 introduce dependent variables into conjunctions. Here the conclusion states the *existence* of an anonymous term that is in the extension of both S and P.

$\{M \rightarrow P,\ M \rightarrow S\}$	$\vdash$	$(\#x() \rightarrow P) \wedge (\#x() \rightarrow S)$	$<F_{exi}>$

Table 5.13: Sample Dependent-Variable Introduction Rule

This inference can only produce positive conclusion, and the conclusion reaches maximum confidence when both premises are absolutely true. Therefore, the truth-value function is the following:

$$F_{exi}: \quad f = 1, \quad c = and(f_1, c_1, f_2, c_2)$$

The reverse of the rule in Table 5.13 can be seen as a special type of unification to match a dependent variable with a constant, as given in Table 5.14. The truth-value function of the abduction rule is used here, because the conclusion gets evidence only when the first premise is positive, in which case term M is compared to the anonymous term

$\#x()$. Under the condition of $M \rightarrow S$, if $(\#x() \rightarrow P) \wedge (\#x() \rightarrow S)$, then $M \rightarrow P$ looks more likely, otherwise less likely.

$\{M \rightarrow S,\ (\#x() \rightarrow P) \wedge (\#x() \rightarrow S)\}\ \vdash\ M \rightarrow P\ \ <F_{abd}>$

Table 5.14: Sample Dependent-Variable Elimination Rule

The rules in Table 5.13 and Table 5.14 are only about the extension of S and P. Similarly, there are rules that only process the intension of the terms involved. As required before, in NAL a dependent variable is only introduced into a conjunction, and an independent variable into both sides of an implication or equivalence.

Variables can be introduced into statements where other variables exist. When an independent variable is introduced, the existing dependent variables become its function, until it is unified with a constant. The rules for multiple variables in Table 5.15 can be extended to handle more than two variables. Please note that here the four conclusions correspond to the four sentences in the previous "lock-key" example. It shows that NAL has rules to produce all meaningful combinations of variable terms.

The revision rule is also extended to unify variable terms. For example, statements $(\#x \rightarrow S) \Rightarrow (\#x \rightarrow P)$ and $(\#y \rightarrow S) \Rightarrow (\#y \rightarrow P)$ can be merged together.

$\{(\#x \rightarrow P) \Rightarrow (M \rightarrow (\perp\ R\ \#x\ \diamond)),\ M \rightarrow S\}$ $\vdash ((\#y \rightarrow S) \wedge (\#x \rightarrow P)) \Rightarrow (\#y \rightarrow (\perp\ R\ \#x\ \diamond))\ \ <F_{ind}>$
$\{(\#x \rightarrow P) \Rightarrow (M \rightarrow (\perp\ R\ \#x\ \diamond)),\ M \rightarrow S\}$ $\vdash (\#y() \rightarrow S) \wedge ((\#x \rightarrow P) \Rightarrow (\#y() \rightarrow (\perp\ R\ \#x\ \diamond)))\ \ <F_{exi}>$
$\{(\#x() \rightarrow P) \wedge (M \rightarrow (\perp\ R\ \#x()\ \diamond)),\ M \rightarrow S\}$ $\vdash ((\#y \rightarrow S) \Rightarrow ((\#x(\#y) \rightarrow P) \wedge (\#y \rightarrow (\perp\ R\ \#x(\#y)\ \diamond)))\ \ <F_{ind}>$
$\{(\#x() \rightarrow P) \wedge (M \rightarrow (\perp\ R\ \#x()\ \diamond)),\ M \rightarrow S\}$ $\vdash (\#y() \rightarrow S) \wedge (\#x() \rightarrow P) \wedge (\#y() \rightarrow (\perp\ R\ \#x()\ \diamond))\ \ <F_{exi}>$

Table 5.15: Sample Multi-Variable Introduction Rules

5.2.3 Hypothetical inference

As described above, a variable term is a special term which can be used as a *symbol* of other terms. By "symbol," it is meant that the meaning of a variable term is not fully grounded on the experience of the system, until it is "bounded" to a (non-variable) term. The same symbol can be used to stand for different terms in different statements. It is similar to pronouns in natural languages, such as "it," "that," or "everything."

Variable terms give NARS the capability of using the same term to indicate different concepts, or with different "interpretation." This is needed in hypothetical inference.

For example, conditional statement

$$(((\{\#x\} \times \{\#y(\#x,\ \#z)\}) \rightarrow parent) \wedge ((\{\#y(\#x,\ \#z)\} \\ \times\{\#z\}) \rightarrow brother)) \Rightarrow ((\{\#x\} \times \{\#z\}) \rightarrow uncle)$$

can be seen as a "rule" by which the system derives "$(\{Mary\} \times \{Tom\}) \rightarrow uncle$" ("Tom is the uncle of Mary") from "$(\{Mary\} \times \{Joe\}) \rightarrow parent$" ("Joe is the parent of Mary") and "$(\{Joe\} \times \{Tom\}) \rightarrow brother$" ("Tom is the brother of Joe"). However, accurately speaking, the inference rule used is the deduction rule, with the above "rule" (the conditional statement) as a premise.

In this way, what is usually called a "rule" in rule-based systems is formalized in NARS in two different levels. Conceptually, a rule indicates that a certain statement can be derived from certain other statement(s). In NARS, a small set of such rules are actually formalized as procedural inference rules, which are part of the logic. The other "rules" are formalized as implication and equivalence statements, which, when used with the above deduction rule, will effectively work as an inference rule. The former group includes rules defined on the built-in logical relations (*inheritance*, *similarity*, *implication*, and *equivalence*), while the latter group includes "rules" on the other acquired "ordinary" relations (such as *parent*, *brother*, and *neutralization*).

Consequently, NAL can be used as a meta-logic of an arbitrary logic, by representing the rules of that logic as NARS implication/equivalence statements. For example, we can "emulate" first-order predicate logic (FOPL) in NARS in this way. When implemented in a computer system, such an emulation will surely be more complicated and less efficient than a direct implementation of FOPL, but at the same time, it does have the benefit of allowing the system to reason "at the outside" of FOPL. For example, non-deductive inference (induction, abduction, analogy, ...) is invalid within FOPL, though it clearly plays an important role when a human mind uses a logic like FOPL. We often first reason outside the system to get a hypothesis, then inside the system to prove or disprove it.

The above description can be extended from FOPL to other formal or mathematical theory. NARS can embed them as subsystems, and do inference inside and outside, so that the system's empirical experience becomes relevant even when an abstract mathematical theory is thought about.

Another issue related to this is the idea of "local axiomization." As repeatedly mentioned before, the key assumption of NARS is that the system works with insufficient knowledge and resources, so that every judgment is only certain to a degree, and for empirical knowledge, confidence never reaches its maximum value 1, that is, it is always revisable. On the other hand, there are analytic statements, whose truth values are independent to the experience of the system, and is a matter of definition. A mathematical (or other formal) theory consists of a set of analytic statements as axioms, and a set of rules (which can be represented as NARS implications). Consequently, though NARS is non-axiomatic with respect to its empirical knowledge, it can contain axiomatic subsystems. When the system works "within" such a subsystem, no uncertainty is allowed.

How do these two parts interact with each other? It is similar to how the human mind uses mathematics and formal logics to solve practical problems. It contains several steps:

1. For a given task T, the relevant empirical knowledge is collected into a knowledge base K.

2. A formal theory F is selected to solve T given K.

3. An interpretation I is build, which maps T and K into T' and K', which are questions and statements in F.

4. A solution S' is found in F for T', according to K'.

5. A solution S is obtained for T from S', under the interpretation I.

In this process, usually only Step 4 is within an axiomatic system. As a result, to apply a mathematical theory to a practical problem does not give the conclusion the status of a mathematical theorem, and the process is highly biased by the experience of the system. The interpretation I in the above serves the same purpose as variable substitution, by which abstract terms in the formal theory are mapped into concrete terms in the domain. In this sense, all abstract terms are variable terms discussed in this section.

5.3 NAL-7: temporal statements

So far, the truth values of statements in NAL are determined by taking all past experience as relevant. However, sometimes we are only interested in the truth value of a statement at a given time. For these situations, NAL-7 introduces *time* into statements.

5.3.1 Time and events

In NAL, an "event" is defined as a statement whose truth value holds in a certain period of time. As a result, its truth value is time dependent, and the system can describe its temporal relation with other events.

Accurately speaking, almost all empirical statements are time dependent, and few statements are about relations holding forever. However, for practical purposes, it is not always necessary to take the time attribute of a statement into consideration. Therefore, whether a *statement* should be treated as an *event* may change from context to context, and events are just statements whose time attributes are specified. On the contrary, the time interval of a "non-event" statement is unspecified, except that it includes the current moment.

In NAL, time is represented indirectly, through events and their temporal relations. Intuitively, an event happens in a time interval, and temporal inference rules can be defined on these intervals [Allen, 1984]. However, in NAL each event is represented by a term, whose corresponding time interval is not necessarily specified. In this way, NAL assumes less information. When the duration of an event is irrelevant, it can also be treated as a point in the stream of time.

In NAL, the temporal order between two events E_1 and E_2 can be one of the following three cases:

- E_1 happens before E_2 happens,
- E_1 happens after E_2 happens,
- E_1 happens when E_2 happens.

This design is consistent with the observation that among human languages, there are three universal temporal primitives: "before," "after," and "when" [Wierzbicka, 1996]. Obviously, the first two cases correspond to the opposite directions of the same temporal relation. Therefore, the primitive temporal relations in NAL are:

"before": which is irreflexive, antisymmetric, and transitive;

"when": which is reflexive, symmetric, and transitive.

They correspond to the *before* and *equal* relation discussed in [Allen, 1984], respectively.

If the temporal relation between two events is more complicated than these three cases, it is always possible to divide an event into subevents, then describe their temporal relations in detail. For example, we can treat "when E_1 starts" and "when E_1 ends" as separate events. This representation covers all kinds of temporal relations between two events, including the ones discussed in the previous studies of temporal inference: *meets*, *overlap*, *during*, *starts*, and *finishes* [Allen, 1984]. In NAL, these temporal relations (and others) are represented not as logic constants, but as "ordinary relations" (discussed in NAL-4). Similarly, we can introduce a term "duration." If the system knows that "$(t_1 \times t_2) \rightarrow duration$", as well as that "$(t_1 \times t_2)$" and event E happen at the same time, it understands that E begins at time t_1, and ends at time t_2. Unlike in interval-based temporal logics, in NAL terms like t_1 and t_2 are events themselves (though their durations are usually omitted), not accurate measurement of absolute time.

If an absolute time is used to represent the temporal property of an event, then that time can be treated as a special event, and these two events are described as happening at the same time.

5.3.2 Temporal operators and relations

Instead of directly using the above two temporal relations between events by themselves, in NAL they are used in combination with certain other logic constants.

First, "E_1 happens before E_2 happens" and "E_1 happens when E_2 happens" both assumes "E_1 and E_2 happen (at some time)," which is "$E_1 \wedge E_2$" plus temporal information. Therefore, we can treat the two temporal relations as variants of the statement operator "conjunction" ("$\wedge$") — "sequential conjunction" (",") and "parallel conjunction" (";"). Consequently, "(E_1, E_2)" means "E_1 happens before E_2 happens," and "$(E_1; E_2)$" means "E_1 happens when E_2 happens." Obviously, "$(E_2; E_1)$" is the same as "$(E_1; E_2)$," but "(E_1, E_2)" and "(E_2, E_1)" are usually different. As before, these operators can take more than two arguments. These two operators allow Narsese to represent complicated events by dividing them into sub-events recursively, then specifying temporal relations among them.

Compared to the two "temporal conjunctions," the original conjunction "$E_1 \wedge E_2$" can be seen as with a default temporal information that both E_1 and E_2 hold at the same time, that is, "$(E_1; E_2)$," therefore the operator ";" is redundant. The other two statement operators, "$\vee$" and "$\neg$," have no temporal variant.

On the other hand, there are the temporal variants of implication and equivalence. For an implication statement "$S \Rightarrow T$" between events S and T, three different temporal relations can be distinguished:

1. If S happens *before* T happens, the statement is called "predictive implication," and is rewritten as "$S \mathrel{/\!\!\!\Rightarrow} T$," where S is called a *sufficient precondition* of T, and T a *necessary postcondition* of S.

2. If S happens *after* T happens, the statement is called "retrospective implication," and is rewritten as "$S \mathrel{\backslash\!\!\!\Rightarrow} T$," where S is called a *sufficient postcondition* of T, and T a *necessary precondition* of S.

3. If S happens *when* T happens, the statement is called "concurrent implication," and is rewritten as "$S \mathrel{|\!\!\Rightarrow} T$," where S is called a *sufficient co-condition* of T, and T a *necessary co-condition* of S.

Similarly, three "temporal equivalence" (predictive, retrospective, and concurrent) relations are defined. "$S \mathbin{/\!\!\!\Leftrightarrow} T$" (or equivalently, "$T \mathbin{\backslash\!\!\!\Leftrightarrow} S$") means that S is an *equivalent precondition* of T, and T an *equivalent postcondition* of S. "$S \mathbin{|\!\!\Leftrightarrow} T$" means that S and T are *equivalent co-conditions* of each other. To simplify the language, "$T \mathbin{\backslash\!\!\!\Leftrightarrow} S$" is always represented as "$S \mathbin{/\!\!\!\Leftrightarrow} T$," so the copula "$\backslash\!\!\!\Leftrightarrow$" is not actually included in the grammar of Narsese. Furthermore, since by default "$\Rightarrow$" and "$\Leftrightarrow$" assumes the pair of events connected by them happen at the same time (i.e., the current time), the relations "$|\!\!\Rightarrow$" and "$|\!\!\Leftrightarrow$" are redundant, and need not be actually implemented.

Adjectives like "past," "current," and "future" indicate temporal relations between events and "now," taken as a special event. When "now" is omitted in a statement, the temporal relations become temporal operators called "tense." The "current" tense is the implicit default, while the "past" and "future" tenses must be explicitly specified. For a statement S, "S happened" and "S will happen" are represented in Narsese as "$\backslash\!\!\!\Rightarrow S$" and "$/\!\!\!\Rightarrow S$," respectively.

5.3.3 Temporal inference

The inference rules introduced in NAL-7 are variants of the rules defined in NAL-5 and NAL-6. The only additional function of these rules is to keep the available temporal information.

As an example, the following is a deduction rule introduced in NAL-5,

$$\{(\wedge\ MA_1 \cdots A_m) \Rightarrow C,\ A_0 \Rightarrow M\} \vdash (\wedge\ A_0A_1 \cdots A_m) \Rightarrow C \ \ {<}F_{ded}{>}$$

Now it has a variant in NAL-7, as listed in Table 5.16.

Since the logical factor and the temporal factor are independent of each other in the rules, these variants can be obtained by considering the two factors separately, then combining them in the conclusion.

$$\boxed{\{(M, A_1, \cdots, A_m) \mathbin{/\!\!\!\Rightarrow} C,\ A_0 \mathbin{/\!\!\!\Rightarrow} M\} \vdash (A_0, A_1, \cdots, A_m) \mathbin{/\!\!\!\Rightarrow} C \ \ {<}F_{ded}{>}}$$

Table 5.16: Sample Temporal Inference Rule

Before temporal information is introduced, in NARS whenever there are two judgments containing the same statement, it will be taken to mean different evidence about the same relation, and revision will be attempted. With temporal information, however, there is another possible interpretation, that is, as *update*, that is, as caused by a *change* in the environment. In this operation, the new judgment does not merge with the old one, but "pushes it into the past," by adding a "past tense" on it, so as to turn it into a different statement. Update usually causes chain reactions, that is, other updates in derived conclusions, as revision does.

In general, inference rules on tense can be derived from ordinary temporal inference rules, by first adding the term "now" into the premises (so as to turn the tense operators into temporal relations), and finally removing the "now" from the conclusions (so as to turn the temporal relations back into tenses). This procedure is similar to the usage of "virtual condition" (which turns an arbitrary statement into an implication) in NAL-5.

Clearly, inference on temporal implication relations can be used to predict the future and to explain the past. However, they are not the same as what we call "causal relation." We can think of the latter as a special case of the former, with additional requirements attached to the concept of "cause," which are highly context-dependent. That is why in NARS the above temporal implication/equivalence relations are defined as part of the logic, and with fixed meaning, while "cause" is left to be an empirically built relation, which will be learned and changed by the system according to experience and context. Even though, the basic capability of what we usually call "causal inference" is already in the system.[7]

5.4 NAL-8: procedural statements

In NAL-8 procedural interpretation is applied to events to represent operations of the system itself. Consequently, declarative knowledge and procedural knowledge are unified in NAL.

[7]This issue will be discussed in detail in subsection 9.4.2.

5.4.1 Operations and procedural inference

In NARS, *operation* is a special kind of event, which can be carried out by the system itself. Therefore it is system dependent: the operations of a system will be observed as events by other systems.

While relations and events are *declarative* knowledge, operations are *procedural* knowledge, in the sense that the meaning of an operation is not only revealed by what it *says*, but also by what it *does*.

Statement "$(\times\ \{A\}\ \{B\}\ \{C\}) \rightarrow R$" intuitively corresponds to "There is a relation R among (individuals) A, B, and C." If R is an event, it becomes "An event R happens among A, B, and C." If R is an operation, it becomes "To carry out R among A, B, and C." To be consistent with the other part of NAL, for the last case in the following we can also use the format "(R, A, B, C)," so that it is just a compound term defined in NAL, with an operator followed by an argument list.

An operation usually distinguishes input and output among its arguments. When an operation is described abstractly, its input arguments are typically independent variables, and its output are dependent variables. For instance, operation "$(plus, \#x, \#y, \#z(\#x, \#y))$" represents "$x$ plus y is z," where x and y are independent variables (input), and z is a dependent variable (output, as a function of x and y).

The knowledge about operations are usually represented as (temporal or not) implication/equivalence statements, which indicates the conditions, causes, and effects of each operation.

For the above example, if statement "$(\times\ \{A\}\ \{B\}\ \{C\}) \rightarrow sum$" represents "The sum of A and B is C," then the following statements describe the necessary postconditions (i.e., effects) of the operations *plus* and *minus*:

$$\begin{aligned}(plus, \#x, \#y, \#z(\#x, \#y)) \mathrel{/\!\!\!\Rightarrow} ((\times\ \{\#x\}\ \{\#y\}\ \{\#z(\#x, \#y)\})\\ \rightarrow\ sum)\end{aligned}$$

$$\begin{aligned}(minus, \#x, \#y, \#z(\#x, \#y)) \mathrel{/\!\!\!\Rightarrow} ((\times\ \{\#y\}\ \{\#z(\#x, \#y)\}\ \{\#x\})\\ \rightarrow\ sum)\end{aligned}$$

Here we see that the same logical relation (such as "*sum*") may correspond to multiple operations (such as "*plus*" and "*minus*") with different input/output partition among its arguments.

The (sufficient or necessary) preconditions of operations can be similarly specified. For example, to carry out $(divide, \#x, \#y, \#z(\#x, \#y))$, $\#y$ cannot be zero, and this necessary precondition can be represented as

$$(divide, \#x, \#y, \#z(\#x, \#y)) \mathbin{\backslash\!\!\!\Rightarrow} \neg(\#y \leftrightarrow 0)$$

More examples of operational knowledge are in the following list:

- "$((\{\#x\} \times \{\#y\}) \rightarrow r_1) \mathbin{/\!\!\!\Rightarrow} (op_1, \#x, \#y)$" indicates that a sufficient precondition for the operation "op_1" to be performed in two arguments is that there is a relation "r_1" between the two.
- "$(op_1, \#x, \#y) \mathbin{\backslash\!\!\!\Rightarrow} ((\{\#x\} \times \{\#y\}) \rightarrow r_2)$" indicates that a necessary precondition for the operation "op_1" to be performed in two arguments is that there is a relation "r_2" between the two.
- "$(op_1, \#x, \#y) \mathbin{/\!\!\!\Rightarrow} ((\{\#x\} \times \{\#y\}) \rightarrow r_3)$" indicates that a necessary postcondition of the operation "op_1" is that there will be a relation "r_3" between its two arguments.
- "$(((\{\#x\} \times \{\#y\}) \rightarrow r_4),\ (op_2, \#y, \#z)) \mathbin{/\!\!\!\Rightarrow} ((\{\#x\} \times \{\#z\}) \rightarrow r_5)$" indicates that after event "$(\{\#x\} \times \{\#y\}) \rightarrow r_4$" happens, executing "$(op_2, \#y, \#z)$" will cause the event "$(\{\#x\} \times \{\#z\}) \rightarrow r_5$" to happen.
- "$((op_3, \#x, \#y),\ (op_4, \#x, \#y),\ (op_4, \#x, \#y)) \Leftrightarrow (op_6, \#x, \#y)$" indicates that executing "op_6" on two arguments is equivalent to the sequential execution of "op_3," "op_4," and "op_5" on them.

What is expressed in these examples is quite similar to what is achieved by logic programming, except that in NARS the logic is fundamentally different from first-order predicate logic, and each statement has a truth value attached to indicate its uncertainty.

Under AIKR, in NARS the preconditions (restrictions) and postconditions (consequences) of an operation are never exhaustively specified. Instead, the system's beliefs on operations only reflect its (limited) experience on them. It is quite possible that certain conditions or consequences, though they exist, never become known to the system. Designed in this way, NARS takes a unusual position toward the well-known Frame Problem [McCarthy and Hayes, 1969], by accepting it as an inevitable consequence of AIKR.[8]

NARS will be implemented with certain primitive operations (both internally-oriented and externally-oriented) exposed to the inference engine, and they can be used as components to build compound operations. Both primitive and compound operations can be called from the inference engine. The system has beliefs, either built-in initially or acquired through experience, about these operations.

Not all operations in such a system are involved in reasoning in this way. NARS has a mixture of *deliberative* and *automatic* processes. The former consists of the system operations visible to the inference engine, but the latter is invisible. To make an operation exposed to the inference engine in this way usually makes its execution more flexible, but at the same time decreases the efficiency, and introduces various kinds of risks caused by the uncertainty in inference.

If an operation is accessible to the inference engine, it will be named by a Narsese term, and so will its (input and output) arguments (some of them may be variable terms). The system's beliefs about the operation will be represented by (temporal) implication and equivalence statements, as shown previously. Furthermore, inference on these statements will incrementally reveal its preconditions and postconditions, as well as its relation with other operations.

Since operations are just events under a procedural interpretation, the inference rules of NAL-8 are the same as NAL-7. The compound operations formed in the inference process correspond to "skills" (procedures) learned by the system. For example, initially the system may only have beliefs about operations "*op*1," "*op*2," and "*op*3" in isolation. By inference, the system will form beliefs on what may happen if the three are executed in a sequence — that may achieve a more

[8]Of course, it does not mean to do nothing at all about it. For a given operation, in NARS there are usually many judgments about its conditions and consequences. In the long run, however, only some of them will be kept, while most of the others will be forgot. For example, if a judgment omits an important condition, it will lead to many failures, and become useless; if a judgment mentions too many conditions, which are usually true, it will become unnecessarily complicated, and lost in resource competition to the simpler-but-equally-effective ones. The situation about consequence is similar. This issue will become more clear in Chapter 6, 11, and 12.

complicated consequence. Of course, due to the insufficiency of knowledge and resources, these derived beliefs may conflict with new evidence and get revised. Even so, after a while, the system will learn various skills, which are compound operations for which the system has beliefs. This process is similar to the "chunking" process in Soar [Newell, 1990], though in NARS it follows a very different logic.

5.4.2 Goals and desire values

In NAL-1, two types of sentences are defined: *judgment* and *question*, where the former is a statement with a truth value, and the latter is a statement whose truth value should be determined by the system. In NAL-8, the third type of sentence, *goal*, is introduced.

In Narsese, a goal has a format similar to a question, that is, it is a statement without a truth value attached, and may contain variables to be instantiated. However, their semantics are different. While a question asks the system to evaluate the truth value of the statement (and maybe find constant terms for the variables) by *inference* only, a goal asks the system to carry out some *operations* to make the statement true — of course, given the inevitable uncertainty in the consequence, it actually means "to make it as close to truth as possible."

NARS usually has multiple goals, and they may conflict with one another, in the sense that the achieving of a goal makes another one harder to be achieved. Therefore, the system must from time to time make decisions about whether to pursue various goals or whether to take various operations.

Since the conflicting goals may not be directly related to each other, we cannot expect the system to have explicit knowledge about how to handle every possible conflict. Instead, a common solution in the study of decision making is to define a "degree of desire," or *desire value*, on goals, so that every pair of goals is comparable.

Under AIKR, in NARS we cannot take the desire value of a goal as a known constant, as in most decision-making theories. Instead, it is something that the system needs to find out by inference. Since the desire value of a goal G is determined according to the system's experience, the truth value of statement is used to represent this degree of desire. To do that, a virtual statement D is introduced for the "desired state.” Like the virtual statement E used for “all evidence” in NAL-5 or the virtual event “*now*” for tense definition in NAL-7, D does not appear within the system, but is used in the meta-theory to design related rules.

With the help of D, the desire value of a goal G is defined as the truth value of statement “$G \Rightarrow D$,” that is, the degree that the desired state is implied by the achieving of this goal.

In this way, the desire-value functions can be derived from the truth-value functions. For example, if goal G has desire value d, and the system believes that G can be achieved by operation A (with truth value t), then the system has premises “$G \Rightarrow D <d>$” and “$A \Rightarrow G \ <t>$.” From them, by deduction the system gets “$A \Rightarrow D < F_{ded} >$,” which means that A becomes a derived goal, with a desire value obtained by using the deduction truth function with d and t as arguments. After dropping the virtual statement D, in the system we get a rule that derives a potential goal A (with a desire value) from a goal G (with a desire value) and a belief “$A \Rightarrow G$” (with a truth value).[9]

Now we can attach a desire value to every statement in the system, because it may become a goal in the future, if it is not already a goal. This value shows the system’s “attitude” about the situation in which the statement is true.[10]

5.4.3 Goal-related inference

When there is no operation that can directly achieve a given goal, the system will do inference to indirectly achieve it. In NAL-1, we have seen how backward inference is used to derive questions. For all the rules introduced in NAL-2 to NAL-7, the same isomorphism holds between forward and backward inference, so all the rules defined in them can also be used to derive questions. A similar situation happens to goals.

[9]This treatment is directly related to the previous philosophical discussion on “desire as belief” [Lewis, 1988]. However, in NARS, a desire is not reduced into a belief. In the system, a belief on statement S and a desire for S to be true are clearly distinguished from, though also closely related to, each other.

[10]This desire value will eventually be attached to every term, to represent the system’s “feeling” about it. If the term is not a statement, its desire value will be determined by the beliefs in which it appears.

When a goal G and a judgment J are taken as premises in an inference step, the judgment may provide a direct solution to the goal, if its truth value indicates that the goal has already been somehow realized (so nothing needs to be done). Otherwise, a derived goal G' can be produced, if and only if G can be derived from J and G'.

The backward inference on goals is related to *planning*. For a given goal, the inference engine can find a group of operations, organized by the "," and ";" operators defined in NAL-7, that achieve the goal (i.e., to make it true as the consequence of the execution of the operations). By executing the plan, and adjusting it when necessary, the internal or external environment is changed to turn the goal into reality. When repeatedly appearing groups of operations are memorized, repeated planning is avoided, and the system learns a new skill.

The planning process in NARS is different from what is usually called "planning" in AI, where the process starts with a goal and a set of primitive operations. In NARS planning incrementally builds complicated plans (i.e., procedural statements), and accumulates knowledge about them. It is just in rare cases that the system starts from the primitive operations, and goes all the way to achieve a complicated plan.

If an operation A will contribute to the achieving of goal G_1 but makes goal G_2 less likely to be achieved, the system will contains judgments "$A \Rightarrow G_1 < f_1, c_1 >$" and "$A \Rightarrow G_2 < f_2, c_2 >$," where f_1 is near 1, and f_2 is near 0. When both G_1 and G_2 are desired (to different degrees), the system will get two "$A \Rightarrow D$," that is, the goal A gets two desire values, obtained from different sources. Clearly, they should be merged using the revision rule.

Since the above process may be repeated when other goals are taken into consideration, after a while the desire value of a goal is usually influenced by many other goals. The final decision of executing an operation is made when the expectation of the desire value of the goal is above a certain threshold, m ($m > 0.5$), which is a system parameter (like the k defined in NAL-1). That is, the system only pursues goals whose overall expected desire value is sufficiently high.

As we have seen, the decision making procedure in NARS is specified differently from the conventional decision-making research (such as [Savage, 1954, Jeffrey, 1965]), in the following aspects:

- A goal is not a state, but a statement — under AIKR, in NARS no statement (within a system) can completely describe a state (of the environment).

- The desire value of a statement may change over time when new information is taken into consideration.

- The likelihood of an operation to achieve a goal is not specified as a probability, but as a truth value defined in NARS. In a truth value, there are two independent factors: the frequency value is similar to probability, and the confidence value is similar to the "third dimension" in decision making (beside desirability and likelihood) suggested by Ellsberg, that is, "a quality depending on the amount, type, reliability and 'unanimity' of information, and giving rise to one's degree of 'confidence' in an estimate of relative likelihoods" [Ellsberg, 1961].

- The decision is on whether to pursue a goal, but not on which goal is the best in a complete set of mutually-exclusive goals (or, as a special case, operations) — under AIKR, such a set cannot be obtained.

There are still similarities between this approach and traditional decision theory. Both approaches let decisions be based on quantitative comparison between alternatives, and in the measurement both the degree of desire of a goal and the likelihood that the goal will be achieved by an operation are involved.

Of course, the above discussion only provides the fundamentals of decision making, and there are advanced issues to be addressed in the future, such as the timing of operation execution and observation of the result of operation.

An important point about NAL-8 is that since some of its statements are actually system operations, the results of the inference process are no longer fully expressible in Narsese. That is, in the previous layers of NAL, no matter what inference rules are used, both input and output of the process are Narsese sentences, that is, the system does nothing else but generate new sentences from given sentences. After NAL-8 is implemented, this is no longer the case. Operations may have "side effects" that go beyond Narsese. As a simple example, if an operation converts its sole argument from a Narsese term into an ASCII string, then prints it out in the default printer, then it is something that the inference engine can control, but some of its effects happen in the physical world (e.g., as ink marks on a piece of paper). Similarly, some operations may be triggered by physical signals, and as a result, convert the signals into Narsese judgments. In this way, beside other things, NAL-8 provides an interface between NARS (as an inference engine) and various sensorimotor mechanisms.

5.4.4 NAL summary

Since NAL-8 is the last extension of the NAL system, now is the time to summarize NAL, as defined from NAL-1 to NAL-8.

The complete grammar rules of Narsese are listed in Table 5.17.

There are some additional notes about the Narsese grammar:

- Confidence values 0 and 1 are used in the meta-language of Narsese only, and cannot appear in actual sentences in the system.

- Many term operators can also be used in the "infix" form, that is, between components.

The semantics of Narsese has been introduced for Narsese-1. The later extensions of Narsese do not change the principle by which truth value and meaning are defined for the language, but allow other types of copula (beside inheritance) to be involved when determining the truth values of statements, as well as when determining the meanings of terms. Furthermore, the meanings of terms are also determined by the compositional relations between compounds and components.

The inference rules of NAL has been introduced in the inference tables in the previous chapters. However, these tables are by no means complete, for the following reasons:

- Many compound-related rules have variants, by changing the number and location of the involved components. Only certain representative cases of these rules are listed in the tables.

<*sentence*>	::=	<*judgment*> | <*question*> | <*goal*>
<*judgment*>	::=	<*statement*><*truth-value*>
<*goal*>	::=	<*statement*><*desire-value*>
<*question*>	::=	<*statement*>?
		| ? <*copula*><*term*> | <*term*><*copula*>?
<*statement*>	::=	<*term*> | (<*term*><*copula*><*term*>)
		| ($\neg$ <*statement*>)
		| ($\wedge$ <*statement*><*statement*>$^{+}$)
		| ($\vee$ <*statement*><*statement*>$^{+}$)
		| (, <*statement*><*statement*>$^{+}$)
		| (<*word*> <*term*>$^{+}$)
		| (<*tense*> <*statement*>)
<*term*>	::=	<*word*> |(<*statement*>)| <*variable*>
		| {<*term*>$^{+}$} | [<*term*>$^{+}$]
		| ($\cap$ <*term*><*term*>$^{+}$) | ($\cup$ <*term*><*term*>$^{+}$)
		| ($-$ <*term*><*term*>) | ($\ominus$ <*term*><*term*>)
		| ($\times$ <*term*><*term*>$^{+}$)
		| ($\perp$ <*term*><*term*>* $\diamond$ <*term*>*)
		| ($\top$ <*term*><*term*>* $\diamond$ <*term*>*)
<*variable*>	::=	# <*word*> | <*variable*> (<*variable*>*)
<*copula*>	::=	$\rightarrow$ | $\leftrightarrow$ | $\circ\!\!\rightarrow$ | $\rightarrow\!\!\circ$ | $\circ\!\!\rightarrow\!\!\circ$
		| $\Rightarrow$ | $\Leftrightarrow$ | $/\!\!\!\Rightarrow$ | $\backslash\!\!\!\Rightarrow$ | $/\!\!\!\Leftrightarrow$
<*tense*>	::=	$/\!\!\!\Rightarrow$ | $\backslash\!\!\!\Rightarrow$
<*truth-value*>	:	a pair of real number in $[0,1] \times (0,1)$
<*desire-value*>	:	the same as <*truth-value*>
<*word*>	:	a string in a given alphabet

Table 5.17: The Complete Grammar of Narsese

- As explained in NAL-5, the meta-level implication relations between statements can be used as inference rules, which are not listed in the inference tables.
- The inference rules of NAL-7 and NAL-8 have not been finalized.

Even so, these tables are quite comprehensive in representing the inference rules of NAL.

As mentioned previously, a "logic" consists of a language, a semantics, and a set of inference rules. Now we have seen these parts of NAL, though we haven't fully discussed the nature of this logic, or how to implement it in a computer system. That is what the following chapters will do.