ADVANCES IN KNOWLEDGE ENGINEERING

知识工程进展 1988

第二届全国知识工程研讨会论文选集

中国地质大学出版社出版

ADVANCES IN KNOWLEDGE ENGINEERING

知 识 工 程 进 展 (1988)

中华人民共和国第二届全国知识工程研讨会论文选集

孙 怀 民　　主 编

中国地质大学计算机科学系资助
中国地质大学出版社出版
武 汉　　1988年12月

内 容 提 要

《知识工程进展(1988)》系《计算机研究与发展》编辑部和第二届全国知识工程研讨会组织、程序与学术委员会联合编辑的论文选集.文集中共收入四十多篇论文，包括五篇特邀与专题报告论文，内容涉及人工智能与知识工程的十几个方向，主题是以知识为基础的系统，侧重工程实现，比较集中地反映了国内该领域的最新研究成果和国内外研究发展动态.对于了解和分析国内外已有的研究工作，避免低水平的重复研究有十分重要的参考价值.部分文章可作为补充教材.

读者对象：计算机软硬件、自动化、信息处理等专业的工程技术人员，大学教师、研究生和本科高年级学生.

知 识 工 程 进 展 (1988)

China University Of Geosciences Press

主　　编　　孙怀民

责任编辑　　陈绪诚、于桂芝等

*

中国地质大学出版社出版

405 印刷厂印刷　湖北省新华书店经销

*

开本 787×1092 1/16 印张 23 字数 569 千字(照排胶印)

1988 年 12 月第 1 版　　1988 年 12 月第 1 次印刷

ISBN 7-5625-0193-9/TP·8

印数 1—3000 册

定价：7.85 元

序　　言

古往今来，艺苑的文人最喜欢的是春天，在许多传世之作的字里行间，随处可见对春天的赞美.春天的美，在于她似意境深远的诗，似联想无穷的画，洋溢着生命，充满了希望.春天固然美，秋天更迷人.每当秋天到来之际，那些在厚实的土地上曾辛勤地劳作、耕耘的人们，静静地企盼着田间的硕果，金色的田野——那醉人的丰收之景出现.知识工程界也一样.

终于，《知识工程进展（1988)》出版了，科学园地又添了一棵新苗.多少年了，人们希冀能在召开全国性学术会议的同时，象国际会议一样出版论文集，迅速报道国内科研成果.然而，由于国家的贫穷，无数次努力均告失败.感谢中国地质大学及其出版社和计算机科学系对我们的信任和支持，同时也要感谢为论文集的出版付出了辛勤劳动的所有同志，没有他们的努力，就不可能有论文集的诞生.

智能软件学组是中国计算机学会软件专业委员会的一个严肃的工作小组，在主持全国知识工程会议的审稿过程中，制定了严肃、科学、公正的审稿程序，包括设立申辩、终审制度，基本上做到了对所有投稿者不问年龄、学历、来自何处，论文评审一个标准，一视同仁.虽然如此，审稿仍有可能遗漏非常优秀的论文，衷心欢迎大家公开评论，以利于改进我们的工作.

这次会议收到的近130份征文中，有十几篇论文因未按征文要求撰写和寄到，根据会议制定的审稿程序，作了退稿处理，其中有的是获奖成果，这是很可惜的.

《知识工程进展（1988)》的出版，是中国知识工程学界走向世界的又一个起点.科学是美丽的，新兴的学科总是吸引了一批最富有创造性和幻想的人，历史赋予了中国知识工程学界全体工作者艰巨的使命，任重道远，衷心希望在两年以后的第三届全国知识工程研讨会论文选集中，有更多优秀的论文出现!

孙怀民

1988.8.15

第二届全国知识工程研讨会会议组织机构

第二届全国知识工程研讨会会议主席: 何志均 (浙江大学 . 教授)

第二届全国知识工程研讨会组织、程序与学术委员会

主席:

孙怀民 (北京航空航天大学.教授)

委员:

何志均 (浙江大学 . 教授)　　王树林 (中国科学院计算技术研究所 . 研究员)

管纪文 (吉林大学 . 教授)　　刘尊全 (中国科学院计算中心. 副研究员)

许卓群 (北京大学 . 副教授)　　陆汝钤 (中国科学院数学研究所 . 研究员)

石纯一 (清华大学 . 副教授)　　陶葆兰 (华中理工大学 . 副教授)

陈世福 (南京大学 . 副教授)　　墙芳蹋 (中国地质大学 . 副教授)

蔡经球 (厦门大学 . 副教授)　　刘椿年 (北京工业大学 . 副教授)

陆汝占 (上海交通大学 . 副教授)　胡运发 (国防科技大学 . 副教授)

李应潭 (中国科学院沈阳自动化研究所. 副研究员)

于桂芝 (《计算机研究与发展》编辑部主编,中国科学院计算技术研究所 . 副研究员)

秘书:

赵致琢 (中国科学院数学研究所 . 研究生)

《知识工程进展(1988)》编辑小组

主编: 孙怀民

组员: 孙怀民　刘尊全　许卓群　石纯一　刘椿年　刘大有　于桂芝　赵致琢

关于《知识工程进展(1988)》文集的出版说明

1987年10月7日智能软件学组在泰安举行学组成员会议.与会者根据第一届全国知识工程研讨会会议情况和国内知识工程研究的进展，认为有必要也有条件在今后召开全国知识工程研讨会时编辑会议论文选集，向国内外公开发行，以便及时迅速地向学术界报道和介绍中国知识工程研究的最新成果和国内外知识工程研究动态.1988年春天，学组的这一设想得到了中国地质大学，该校出版社和计算机科学系，《计算机研究与发展》编辑部的支持.3月11日，学组成立了第二届全国知识工程研讨会组织、程序与学术委员会，并对论文选集的审查和编辑程序作出了明确规定.

《知识工程进展(1988)》系按照高级学术刊物(学报级)的审稿程序在二审的基础上，由第二届全国知识工程研讨会组织、程序与学术委员会召开联席审稿会遴选后编辑出版.根据论文题目与审稿人研究方向一致的原则，每篇稿件同时由非同一单位的两名学者分别审稿，使用统一的稿件审查意见书和稿件审查通知书，委员会秘书负责对审稿人姓名严格保密.《知识工程进展(1988)》的入选论文由作者按审稿意见修改，经文集专题编辑复审编辑后付印.学组挂靠单位——中国科学院数学研究所投寄的论文由孙怀民先生按审稿程序组织审查并保存稿件审查意见书.为了发扬学术民主的精神，严肃公正地把好质量关，真正做到对每一位投稿者的稿件认真负责，会议组织、程序与学术委员会设立了申辩终审制度.国内许多学者参加了论文选集的评审工作.虽然如此，会议论文集审稿仍有可能遗漏非常优秀的论文，我们欢迎大家公开评论.

《知识工程进展(1988)》入选论文的标准是:

1. 理论与实践相结合，侧重工程实现，反映我国知识工程学界的最新研究成果;

2. 所有入选论文应不低于通报级杂志的水平，其中，至少三分之一的论文应达到学报级杂志的水平.大会30分钟报告的论文应达到学报级杂志的水平.

第二届全国知识工程研讨会会议录用论文分大会30分钟报告，分组15分钟报告和不报告交流论文.其中，30分钟报告和15分钟报告论文选入文集.这次论文评审工作由于贯彻执行了严肃，认真，科学，公正的审稿程序，使论文审稿意见具有了相当的可信度和权威性.学组相信，公开审稿程序和说明审稿过程将是有益的.同时，我们愿借此机会向所有支持学组工作和第二届全国知识工程研讨会的单位及个人表示衷心的感谢.

1988年6月11日由第二届全国知识工程研讨会组织.程序与学术委员会召开的联席审稿会根据《知识工程进展(1988)》主编孙怀民先生的建议，决定成立编辑小组(名单见前页)，负责论文修改稿的复审和编辑.编辑小组在主编的领导下认真开展了工作，全体成员对论文集的审查和编辑负有不可推卸的责任.

《计算机研究与发展》编辑部
第二届全国知识工程研讨会
组织、程序与学术委员会
1988.7.30.北京

目　录

类型理论——人工智能的新工具

ALT 语言的形式定义及应用

李　未

(北京航空航天大学)

摘要：本文介绍了类型理论的进展，剖析了用类型理论处理问题的一般方法.作为将现代类型理论推向实用的一个尝试，本文定义了语言 *ALT*，它是一个函数式语言，除了具有此类语言的结构和特点外，它还具有一套类型系统，引入了∑与∏类型和类型算子与∑及∏超类型，因此 *Martin-löf* 及 *ELF* 等类型理论均可看成此类型系统的子理论.本文给出了 *ALT* 的结构操作语义，通过几个例子说明了 *ALT* 语言在描述人工智能中高层次概念方面的作用.

一、引　言

类型是程序设计语言的一个重要组成部分，有关类型的研究和实现一直是程序理论研究一个活跃而重要的分支.自 Martin-löf 于七十年代末期提出他的直觉主义类型理论之后，有关类型理论及应用的研究便进入了一个新阶段.当前的现状是:

一方面新的类型和与其相应的理论不断涌现，例如，法国 Coquand 及 Huet 的构造理论（参见〔Coquand Huet85〕），英国 Plotkin 等人的 ELF（Edinburgh Logic Framework）[8]等.这些理论的出现使我们有可能形式化地描述在软件工程及人工智能中常遇到的诸如"命题"，"公则"，"变换"，"问题"，"证明策略"等这类高阶概念.通常的软件开发环境是只描述特定的"命题"，具体的"规则"等等，都不能刻划这些抽象的概念.在这些类型理论进展的启发下，新的程序开发方法也不断出现，Constable 等人根据 Martin-löf 类型理论给出通过对功能描述进行证明而构造出与描述一致的程序的方法〔6〕，以及爱丁堡大学使用 ELF 构造交互式的证明编辑器的方法〔16〕，总之，类型理论的最新发展使我们有可能以新观点来看待和构造软件工程及人工智能的开发环境和工具.

另一方面，如果我们对现有的几个类型理论进行更深入的分析和研究，就会发现它们都是为某些特定的目的服务的.例如 Matin-löf 的类型理论是为了解释和实现直觉主义的一阶逻辑而建立的，他引入了∏类型和∑类型以便实现全称量词和存在量词，他的理论没有涉及有关超类型（Kind）的概念.爱丁堡的 ELF 是为了解释和实现各种逻辑系统而建立的〔1〕，它引入了类型算子及超类型的概念以便描述和实现逻辑系统中的推理规则.可是，在 ELF 中又缺少∑类型及∑超类型.因此从软件工程的角度来看，首先上述这些工作只是给出了理论的框架，它们还不是程序设计语言，其次这些理论都过于狭窄，不能以它们中的任意一个作为框架来设计软件开发工具.

本文是将现代类型理论推向实用阶段的一个尝试.文中将介绍一个程序设计语言 ALT (An Applicative Language with a Powerful Type System) .它是一函数式程序设计语言，具有函数式语言的特征和结构（参见〔Milner.87〕）.ALT 又是根据一个更为一般的类型理论而设计的，它将把 Martin-löf Coquand & Huet 以及 ELF 的类型理论包括在其中作为子理论.与本文比较接近的工作是 Cardelli 设计的语言 Quest，但 ALT 比 Quest 具有更强的表达能力.

本文将使用结构操作语义方法给出 ALT 的形式描述，即用抽象语法定义 ALT 的语法，用公理及规则定义 ALT 的静态及动态语义.结构操作语义方法是根据 λ 演算的形式化描述引伸出来的〔10〕，但它较后者更为灵活有更强的表达能力，此外本文还将通过一些例子说明 ALT 的应用.

本文的安排如下：第二节讨论类型理论的一般方法，引出设计 ALT 的必要性，第三节给出 ALT 的抽象语法，第四、五及六节讨论 ALT 的静态，动态语义和两种语义的交互作用，第七节给出 ALT 在描述近似推理等方面的应用.

二、类型理论的一般方法

我们将通过两个例子来考察类型理论处理问题的一般方法.

例 1. Martin-löf 的类型理论及直觉主义逻辑

Martin-lof 的目标是解释和实现直觉主义的一阶逻辑，他的方法可以概括为下述三个步骤：

a. 将命题解释成类型，当我们陈述一个命题时，例如"张三是个贼"，我们实际上做了一判断（judgement）.因此，命题即判断，命题为真也就是判断有证据（evidence），所以命题（判断）可以看成其全部证据的集合.对于数学命题，其证据就是关于此命题的证明，因此命题是全部证明的集合.对于计算机科学，我们不讨论一般的集合，只讨论能行可构造的集合，这就是类型（type）.命题是其全部证明构成的类型.

b. 确定命题是哪些类型及怎样构造它们，这就是 Martin-löf 的类型理论，他引入的类型为，如果 A 及 B 为类型，则类型的卡氏积 $A\times B$，类型的和 $A+B$，函数空间 $A\to B$，$\prod$类型$\prod_{X:A}\cdot B$（无穷乘积）及$\sum$类型$\sum_{x:A}\cdot B$（无穷和），Martin-löf 给出了构造这些类型的规则〔13〕.

c. 用已定义的类型解释命题连接词，这实际上是简单的语法直接翻译（syntax-directed translation）.用 o 表示此由命题到类型的翻译，则有：

$$(A\wedge B)^{\circ}=A^{\circ}\times B^{\circ}$$
$$(A\vee B)^{\circ}=A^{\circ}+B^{\circ}$$
$$(A\supset B)^{\circ}=A^{\circ}\to B^{\circ}$$
$$(\forall_{X:A}.B\ (X))^{\circ}=\prod_{X:A}.\ (B\ (X))^{\circ}$$
$$(\exists_{X:A}.B\ (X))^{\circ}=\sum_{X:A}.\ (B\ (X))^{\circ}$$

例 2. 用爱丁堡 ELF 表示逻辑系统

建立 ELF 的目的是为了解释和实现各种逻辑系统，为此 ELF 不仅要描述逻辑系统中的语法范畴，还应能描述系统中的公理和推理规则.后者是问题的难点.让我们来分析一

个具体的规则，一阶逻辑中关于"非"的消去规则：

$$\sim\sim E:\ \frac{\sim\sim\varphi}{\varphi}$$

其中～～E是规则名.规则的含义是对任意合式公式 φ，由～～φ 真推出 φ 真.如果用 o 表示全体合式公式构成的类型，$\vdash$ 表示推出，则此规则可以写成：

$$\sim\sim\mathrm{E}:\ \wedge_{\varphi:o}.\ (\sim\sim\varphi \vdash \varphi)$$

实际上一个规则的形式是由下述两种基本形式组成的：

1. 假设性形式（hypothetical form）即～～$\varphi \vdash \varphi$，更一般地，$J_1 \vdash J_2$，它表示 J_2 是 J_1 的逻辑结论.从直觉主义的观点看，只有当将 J_1 的证明映射到 J_2 的证明的函数存在时，这种形式才成立，因此可以把假设性形式 $J_1 \vdash J_2$ 等同于定义域为 J_1 的所有证明、值域为 J_2 的所有证明的全体函数构成的集合.

2. 框架形式（schematic form）.如果用 J（φ）代表～～$\varphi \vdash \varphi$，则框架形式为 $\wedge_{\varphi:o} J(\varphi)$，它的含义是对所有命题 φ，形式 J（φ）成立.显然，当存在一个函数使对每一个合式公式 φ 都产生一个 J（φ）的证明时，框架形式才成立.因此框架形式可以看成全体上述函数构成的集合.

进一步地分析告诉我们，这两种规则形式对应的集合，实际上是一种新的类型称为高阶类型或称超类型（kind）.逻辑系统中的每一推理规则都对应于一种特定的超类型，其所以不空是因为有此规则存在，因此可以把规则各作为此规则所对应的超类型的元素.

a. 当我们一旦确定了把规则的形式定义成超类型这一原则之后，下一步工作就是决定这些超类型是什么和怎样构造它们.仍以～～$\varphi \vdash \varphi$ 为例，它是一个集合，其元素是定义域为全体～～φ 的证明，值域为 φ 的所有证明的函数.若用 trne（φ）表示 φ 的全部证明，则 trne（φ）$\in$ Type，其中 Type 是元素为类型的集合它是一个高阶类型，或称超类型.因此，这里 trne 可以看成算子 trne: O→Type.而～～$\varphi \vdash \varphi$ 将等同于函数空间 Trne（～～φ）→trne（φ）.总之 ELF 要处理三种语法范畴即对象（object），类型（type）及高阶类型（kind），有关它们的构造请参见[8].

b. 最后一步是将规则形式用类型及高阶类型表示出来，不难看出这也是一个语法直接翻译，若用 o 表示这个翻译，则 o 可递归定义如下：

$(\varphi)^{\circ} = \mathrm{trne}\ (\varphi)$

$(J_1 \vdash J_2)^{\circ} = J_1^{\circ} \to J_2^{\circ}$

$(\wedge_{\varphi:o} J(\varphi))^{\circ} = \prod_{\varphi:o}{}^{\circ}.\ (J\ (\varphi))^{\circ}$

从上述两个例子的讨论中，我们可以看出类型理论解决问题的一般步骤：

1. 把待解问题用集合的语言描述出来，这里对问题的直觉主义理解常起着重要作用.

2. 能行地构造所需要的集合，我们称这些集合为类型.

3. 用已定义的类型给出问题的精确解释，目前的例子都是一些简单的语法直接翻译，其实更为普遍的将是与环境有关的各种翻译.

上述分析给了我们一个启发，即如果我们能构造一个统一而一般的类型理论，使得 Martin-löf，ELF 等类型理论都是其子理论，以此理论为基础设计一个程序语言，现有的各种类型理论的应用都可以写成这种语言的程序.不难想象，这个语言将为描述软件工程及人工智能中高层次概念提供一个有利的开发工具.

本文介绍的 ALT 就是设计这种语言的一个尝试.首先，它是一个函数式语言，具有函数式语言的一般特点和结构；其次，它有一套很强的类型系统，它不仅包含 Martin-löf 的∑及∏类型，包含 ELF 有关∏类型的算子及∏超类型，还引入了关于∑类型的算子及∑超类型.本文的下述几节将介绍 ALT 的语法，语义及应用.对语言形式定义不感兴趣的读者可以在读完第三节关于 ALT 语言的介绍后，直接跳到第七节读有关 ALT 应用的例子.

三、ALT 的抽象语法

为了更好地理解 ALT 语法成份的语义，本文只介绍它的抽象语法.ALT 由五个语法范畴组成，即对象（object），类型及类型算子（types and operators），超类型（kind），特征序列（signature）及实参序列（argument）.对象的集合构成类型，类型的集合构成超类型.这些语法范畴的 BNF 定义如下:

对象 a::=c | x | fun (S) a | a (D) | 〈D, a〉 |
bind S=a in b | recfun (x:A) a | if b then a else c

类型及算子 A::= C | X | ALL (S) A | SOME (S) A |
FUN (S) A | A (D) | BIND S=A IN B |
〈D, A〉 | IF b THEN A ELSE B

超类型 K::=TYPE | KALL (S) L | KSOME (S) L

特征序列 S::= empty | S, X:K | S, x:A

实参序列 D::= empty | D, A | D, a

对此抽象语法的解释如下:

a，b 等表示对象（包括下标），c 表示常对象，x，y，z 等表示对象变量，fun(S)a 表示多变量带类型的 λ 抽象，a(D)表示 λ 一演算中的实施(application).例如: λx:int, y:int.x 在 ALT 中将表示成 fun (x:int, y:int) x.项〈D, a〉是一多元组，其右端元素 a 可能依赖于 D，即 D 中的变量可能在 a 中自由出现.项 bind S=a in b 是 Martin-löf E 算子的推广，与 Cardelli 的 bind 算子类似[13][4][5].其中，S 是一变量序列，a 是一个分量为对象的多元组，bind 项的含义是将 b 中出现的由 S 定义的变量换成 a 中与此变量对应的项.

我们将用 A，B 等表示类型，C 代表常类型，X，Y 及 Z 代表类型变量.ALL (S) A 是推广了的∏类型，其中 A 可能依赖于 S 中定义的变量，SOME (S) A 是推广了的∑类型.FUN (S) A 是类型的 λ 抽象 (abstraction)，S 中允许有类型变量出现.例如 λx:Type.x 在 ALT 中将写成 FUN (x:TYPE) X，项 A (D) 表示类型函数的实施 (application) .例如: (FUN (x:TYPE) X) (int) 其结果为 int.〈D, A〉是一分量为类型的多元组，右端元素，可以依赖于 D.项 BIND S=Z in B 含义与前面 bind 项类似，但这里的 B 是类型项.

我们用 K，L，M 等表示超类型.TYPE 是一常超类型，其元素为一类型，因此对任意类型 A 有 A:TYPE.超类型 KALL (S) L 的元素是函数，其定义域为 S，而值是包含在超类型 L 中的类型，但 L 可能依赖于 S.超类型 KSOME (S) L 的元素是多元组，其右端元素是一类型且包含在超类型 L 中.

S 将被用来表示特征序列，它代表约束变量及环境.特征序列的元素可以是 x:k，其中 x 是一类型变量，或是 x:A，x 是一对象变量.

我们用 D 表示实参序列，实参序列的元素可以是一个类型 A 或是一对象 a.

例 1. 设 y 是类型为 int 的变量，even (x) 是一函数，当 x 为偶数时其值为真，否则为假 考虑项:

$$a =_{df} \text{if even (x) then 2 else fun (x:int) x}$$

显然 a 的类型是

$$A =_{df} \text{if even (x) then int else ALL (x:int) int}$$

这里 A 的值与 x 的值有关.这样函数 fun (y:int) a 的类型就应为 FUN (Y:int) A，而后者的超类型为 TYPE.

四、静态语义

对任何程序设计语言，如果只根据此语言的 BNF 定义，我们会写出有问题的程序.例如下述程序:

```
program POOR (input, output)
var  x, x: integer;
begin
    x:=y+1
end;
```

这个过程完全符合 BNF 定义，但 x 被说明两次，而变量 y 没有被说明.因此任何语言文本对其语法范畴都要给出一些与上下文有关的要求，这些要求被称为静态语义 (static Semantics) .对类型及超类型的元素的定义也是静态语义的组成部分.本文采用结构操作语义方法定义 ALT 的静态语义，使用这种方法语义可由一组公理和规则给出，满足静态语义的语法成分和程序称为有效的 (valid) 语法成分和程序.我们用

$\vdash S\ s:g$　　表示 S 是一有效的特征序列;

$S \vdash D \approx S_1$　　表示对给定的 S 参量序列 D 的分量个数，顺序和类型与 S_1 一致;

$S \vdash K\ kind$　　表示对给定的 S，K 是一有效的超类型;

$S \vdash A : K$　　表示对给定的 S，A 是超类型 K 的元素;

$S \vdash a : A$　　表示对给定的 S，a 是类型 A 的元素;

令 Ω 代表 ALT 的语法成分，我们定义 FV (Ω) 为 Ω 中自由出现的变量集; DV (S) 为 S 中出现的变量集，S 是一特征序列.

FV 及 DV 可以递归地定义如下:

Signature	empty	S, X:K	S, x:A
DV	Φ	DV (S) $\cup$ {X}	DV (S) $\cup$ {x}

Argument	empty	D, A	D, a
FV	Φ	FV (D) $\cup$ FV (A)	FV (D) $\cup$ FV (a)

Kinds	TYPE	KALL (S) K	KSOME (S) K
FV	Φ	FV (S) $\cup$ (FV (K) $\setminus$ DV (S))	FV (S) $\cup$ (FV (K) $\setminus$ DV (S))

Types	X	ALL (S) A	SOME (S) A
FV	{X}	FV (S) $\cup$ (FV (A) $\setminus$ DV (S))	FV (S) $\cup$ (FV (A) $\setminus$ DV (S))

Types	C	FUN (S) A	A (D)
FV	Φ	FV (S) $\cup$ (FV (A) $\setminus$ DV (S))	FV (A) $\cup$ FV (D)

Types	⟨D, A⟩	BIND S,Y:K = A in B	IF b THEN A_1 ELSE A_2
FV	FV(D)UFV(A)	FV (A) $\cup$ (FV (B) $\setminus$ {DV (S) ,Y})	FV (b) $\cup$ FV (A_1) $\cup$ FV (A_2)

objects	x	fun (S) a	a (D)
FV	Φ	FV(S)$\cup$ (FV(a)$\setminus$DV(S))	FV (a) $\cup$ FV (D)

objects	c	⟨D, a⟩	bind S, y:A = c in b
FV	Φ	FV(D)$\cup$ FV(a)	FV(C)$\cup$ {FV(b)$\setminus$DV(S),y}

objects	if b then a_1 else a_2	recfun (x:A) a
FV	FV (b) $\cup$ FV (a_1) $\cup$ FV (a_2)	FV (A) $\cup$ (FV (a) $\setminus$ {X})

我们还假定 BOOL 类型是 ALT 中预先定义的类型，序列 S_V 递归定义如下

Signature	empty	S_1, X:K	S_1, x:A
S_V	empty	S_{1V}, X	S_{1V}, x

ALT 的静态语义定义如下

特征序列:

(1) $\vdash$ empty sig

(2) $$\frac{S \vdash K \ \text{kind} \quad x \notin DV(S)}{\vdash S,\ x{:}k \ \text{sig}}$$

(3) $$\frac{S \vdash A \ \text{TYPE} \quad x \notin DV(S)}{\vdash S,\ x{:}A \ \text{sig}}$$

实参序列:

(4) $$\frac{S \vdash \text{sig}}{S \vdash \text{empty} \approx \text{empty}}$$

(5) $$\frac{S \vdash S_1 \approx D_1 \quad S \vdash A{:}K \quad x \notin (DV(S) \cup DV(S_1))}{S \vdash S_1,\ x{:}K \approx D_1,\ A}$$

(6) $$\frac{S \vdash S_1 \approx D_1 \quad S \vdash a{:}A \quad x \quad (DV(S) \cup DV(S_1))}{S \vdash S_1,\ x{:}A \approx D_1,\ a}$$

超类型:

(7) $$\frac{\vdash S \ \text{sig}}{S \vdash \text{TYPE kind}}$$

(8) $$\frac{\vdash S_1 \ \text{sig} \quad S,\ S_1 \vdash L \ \text{kind}}{S \vdash \text{KALL}\ (S_1)\ L \ \text{kind}}$$

(9) $$\frac{\vdash S_1 \ \text{sig} \quad S,\ S_1 \vdash L \ \text{kind}}{S \vdash \text{KSOME}\ (S_1)\ L \ \text{kind}}$$

类型及算子:

(10) $$\frac{\vdash S \ \text{sig} \quad X{:}K \ \text{in} \ S}{S \vdash X{:}K}$$

(11) $$\frac{\vdash S_1 \ \text{sig} \quad S,\ S_1 \vdash A{:}\text{TYPE}}{S \vdash \text{ALL}\ (S_1)\ A{:}\text{TYPE}}$$

(12) $$\frac{\vdash S_1 \ \text{sig} \quad S,\ S_1 \vdash A{:}\text{TYPE}}{S \vdash \text{SOME}\ (S_1)\ A{:}\text{TYPE}}$$

(13) $$\frac{\vdash S_1 \ \text{sig} \quad S,\ S_1 \vdash A{:}L}{S \vdash \text{FUN}\ (S_1)\ A{:}\text{KALL}\ (S_1)\ L}$$

(14) $$\frac{S \vdash D \approx S_1 \quad S \vdash A{:}\text{KALL}\ (S_1)\ L}{S \vdash A\ (D)\ {:}L\ [D/S_v]}$$

(15) $$\frac{S \vdash D \approx S_1 \quad S,\ S \vdash A{:}L}{S \vdash \langle D,\ A\rangle{:}\text{KSOME}\ (S_1)\ L}$$

(16) $$\frac{S \vdash A{:}\text{KSOME}\ (S_1)\ L \quad S,\ Z{:}\text{KSOME}\ (S_1)\ L \vdash K \ \text{kind} \quad S,\ S_1,\ Y{:}L \vdash B{:}K[\ (S_1,\ Y{:}L)\ /Z]}{S \vdash \text{BIND}\ \ S,\ Y{:}L = A\ \text{IN}\ B{:}K[A/Z]}$$

(17) $$\frac{S|-b:BOOL \quad S|-A:TYPE \quad S|-B:TYPE}{S|-IF \quad b \quad THEN \quad A \quad ELSE \quad B}$$

对象:

(18) $$\frac{|-S \quad sig \quad x:A \quad in \quad S}{S|-x:A}$$

(19) $$\frac{|-S_1 \quad sig \quad S,\ S_1|-A:L}{S|-fun\ (S_1)\ a:ALL\ (A_1)\ A}$$

(20) $$\frac{S|-a:ALL\ (S_1)\ A \quad S|-D\approx S_1}{S|-a\ (D)\ :a[D/S_v]}$$

(21) $$\frac{S|-D\approx S_1 \quad S,\ S_1|-a:A}{S|-\langle D,\ a\rangle:SOME\ (S_1)\ A}$$

(22) $$\frac{\begin{array}{c}S|-a:SOME\ (S_1)\ A \quad S,\ Z:SOME\ (S_1)\ A|-B:TYPE\\ S,\ S_1,\ Y:A|-d:B[\ (S_1,\ Y:A)\ /Z]\end{array}}{S|-bind \quad S,\ \ y:A=a \quad in\ d:B[a/Z]}$$

(23) $$\frac{S|-A:TYPE \quad S,\ x:A|-a:A}{S|-recfun\ (x:A)\ a:A}$$

(24) $$\frac{S|-b:BOOL \quad S|-a_1:A_1 \quad S|-a_2:A_2}{S|-if \quad b \quad then \quad a_1 \quad else\ a_2:IF\ b\ THEN\ A_1\ ELSE\ A_2}$$

五、动态语义

结构操作语义用定义转移 (transitions) 关系的办法刻划程序的执行，这些转移关系称为语言的动态语义 (dynamic Semantics) [14][11].ALT 的动态语义所定义的转移关系，实际上是带类型的 λ 演算中 β–η 归约的一种推广 (β–η reduction) .由于上节关于静态语义的讨论中已经定义了什么是有效的项，因此本节讨论动态语义时假定所处理的项都是有效的.我们将用下述五种转移关系刻划 ALT 不同语法成分的动态语义:

$S|-K\longrightarrow_k K'$　表示对给定的环境 S，超类型 K 归约为 K′;

$S|-T\longrightarrow_t T'$　表示对给定的环境 S，类型 T 归约为 T′;

$S|-a\longrightarrow_o a'$　表示对给定的环境 S，对象 a 归约为 a′;

$S|-S'\longrightarrow_s S''$　表示对给定的环境 S，特征序列 S′归约为 S″;

$S|-D\longrightarrow_D D'$　表示对给定的环境 S，实参序列 D 归约为 D′;

当上述关系清楚不致混淆时，我们将忽略上述转移关系中箭头下面的 k，t，o，s 及 D，这样五种关系可以归纳地定义如下

超类型:

(1) $$\frac{S_0|-S\rightarrow S'}{S_0|-KALL\ (S)\ L\rightarrow KALL\ (S')\ L}$$

(2) $$\frac{S_0 \mid - L \to L'}{S_0 \mid - \text{KALL (S) L} \to \text{KALL (S) L}'}$$

(3) $$\frac{S_0 \mid - S \to S'}{S_0 \mid - KSOME\ (S)\ L \to KSOME\ (S')\ L}$$

(4) $$\frac{S_0 \mid - L \to L'}{S_0 \mid - KSOME\ (S)\ L \to KSOME\ (S)\ L'}$$

类型及算子:

(5) $$\frac{S_0 \mid - S \to S'}{S_0 \mid - \text{ALL (S) A} \to \text{ALL (S') A}}$$

(6) $$\frac{S_0 \mid - A \to A'}{S_0 \mid - \text{ALL (S) A} \to \text{ALL (S) A}'}$$

(7) $$\frac{S_0 \mid - S \to S'}{S_0 \mid - \text{SOME (S) A} \to \text{SOME (S') A}}$$

(8) $$\frac{S_0 \mid - A \to A'}{S_0 \mid - \text{SOME (S) A} \to \text{SOME (S) A}'}$$

(9) $S_0 \mid - \text{FUN (S) A (D)} \to \text{A } [D / S_V]$

(10) $S_0 \mid - \text{FUN (S) (A } (S_V) \text{)} \to \text{A}$

(11) $$\frac{S_0 \mid - S \to S'}{S_0 \mid - \text{FUN (S) A} \to \text{FUN (S') A}}$$

(12) $$\frac{S_0 \mid - A \to A'}{S_0 \mid - \text{FUN (S) A} \to \text{FUN (S) A}'}$$

(13) $$\frac{S_0 \mid - A \to A'}{S_0 \mid - \text{A (D)} \to \text{A' (D)}}$$

(14) $$\frac{S_0 \mid - D \to D'}{S_0 \mid - \text{A (D)} \to \text{A (D')}}$$

(15) $$\frac{S_0 \mid - D \to D'}{S_0 \mid - \langle D,\ A \rangle \to \langle D',\ A \rangle}$$

(16) $$\frac{S_0 \mid - A \to A'}{S_0 \mid - \langle D,\ A \rangle \to \langle D,\ A' \rangle}$$

(17) $S_0 \mid - \text{BIND S=D IN B} \to \text{B } [D / S_V]$

(18) $$\frac{S_0 \mid - A \to A'}{S_0 \mid - \text{B IN D S=A IN B} \to \text{B IN D S=A' IN B}}$$

(19) $S_0 \mid - \text{IF tt THEN A ELSE B} \to \text{A}$

(20) $S_0 \mid - \text{IF ff THEN A ELSE B} \to \text{B}$