国 际 电 信 联 盟

CCITT

国 际 电 报 电 话 咨 询 委 员 会

黄 皮 书

卷 Ⅵ.8

CCITT高级语言(CHILL)

建议 Z.200

第 七 次 全 体 会 议

1980年11月10－21日 日内瓦

1984年 北京

国 际 电 信 联 盟

CCITT

国际电报电话咨询委员会

黄 皮 书

卷 Ⅵ.8

CCITT高级语言(CHILL)

建议 Z.200

第 七 次 全 体 会 议

1980年11月10—21日 日内瓦

1984年 北京

CCITT图 书 目 录

适用于第七次全体会议（1980年）以后

黄 皮 书

CCITT高级语言（CHILL）
（日内瓦，1980）

黄皮书 卷Ⅵ·8 目 录

1.0 导言

本建议书定义了CCITT高级程序设计语言CHILL。CHILL是CCITT High Level Language的缩写。

CHILL的另一种以严格数学形式表示的定义，发表在CCITT手册中。还有一本《CHILL概述》是CHILL语言的一个引言。

1.1 概述

CHILL主要是为存储程序控制电话交换机的编程而设计的。但它具有极大的通用性，可以应用于其它方面（如报文交换、分组交换、模拟等）。

在设计CHILL时考虑了下列要求（参见1977～1980研究阶段文件：课题8/XI）：

- 能进行大量的编译时检验以提高可靠性；
- 能生成高效率的目标代码；
- 使用灵活、功能强，能覆盖所要求的应用范围，并能发挥各种硬件的作用；
- 利于模块化和结构化的程序开发；
- 容易学习和使用。

CHILL隐含有一个程序开发环境。这个环境除其它功能项外，还包括：分块编译、输入/输出和调试工具。这些功能项在本建议书中没有给出定义。

对于SPC电话交换中已经使用或建议使用的机器类型，CHILL程序能够以独立于机器的形式书写。

CHILL并不打算为上面提到的每个应用领域提供特殊的语言结构，但是有通用的基础结构，为具体应用提供一些合于使用的可能手段。

作为一种语言，CHILL是独立于机器的。但是在具体实现时，可以包含一些实现定义的语言成分。包含这类成分的程序一般是不可移植的。

设计CHILL的前提是要能把它的源码编译成目标代码。设计时并没有考虑到一次扫描编译方法的特殊需要，也没有考虑到使编译程序的体积达到最小。

为了确保安全而又不致于使效率的损失大到不可接受的程度，许多检验可以静态地进行，只有少数语言规则要在运行时检验。违反这些规则将引起运行时的异常。但是，除非程序员自己规定了异常处理程序，这些异常时的运行检验是任选的。

1.2 语言简况

CHILL程序基本上由三部分组成：

- **数据对象**的描述；
- 作用于数据对象的**动作**描述；
- **程序结构**的描述。

数据对象由**数据语句**（说明语句和定义语句）描述，动作由**动作语句**描述，程序结构由**程序结构语句**确定。

CHILL可以处理的数据对象是**值**和存放值的**地点**。动作定义作用于数据对象的操作，以及把值存入地点或从地点中检索值的次序。程序结构确定数据对象的生存期和可见性。

对于在给定上下文中数据对象的使用，CHILL提供了充分的静态检验。

以下各节概述了CHILL中出现的各种概念。每一节介绍与该节同名的一章的内容，概念的详细描述在有关章中说明。

1.3 模式和类

CHILL可处理的数据对象是**值**和可以存放值的**地点**。

每个地点有一个**模式**。地点的模式定义了可以存放于该地点的值的集合，也定义了地点具有的其它性质以及可能存放的值的性质（注意，单靠地点的模式不能确定该地点的所有性质）。地点的性质是：大小、内部结构、只读性、可关联性等。值的性质是：内部表示、次序、可应用的操作等。

每个值有一个类。值的类确定可以存放值的地点的各模式。

CHILL提供下列诸模式范畴：

离散模式 整数、字符、布尔、集合（符号）模式以及这些模式的区段；

幂集模式 某些离散模式的元素组成的集合；

关联模式 约束关联、自由关联以及用作对地点的关联的行；

组合模式 字符串、数组和结构模式；

过程模式 过程看作可处理的数据对象；

样品模式 对进程的标识；

同步模式 用于进程同步和通信的事件模式和缓冲区模式。

CHILL提供一组标准模式的标志。用**模式定义**可以引进由程序定义的模式。这类语言结构有一个所谓的**动态模式**。动态模式是这样一种模式，它的某些性质只能动态地确定。动态模式都是带有运行时参数的参数化模式。非动态模式称为**静态模式**。在CHILL程序中显式标志的模式总是静态的。

在CHILL中，动态模式和类都没有标志。它们只出现在元语言中，用来描述静态和动态的上下文条件。

1.4 地点及其访问

地点是（抽象的）位置，可以把值存进去，也可以从其中取出值来。为了存放或得到一个值，就必须访问一个地点。

说明语句定义用来访问一个地点的名字。它们是：

1. 地点说明；
2. 地点等同说明；
3. 有基说明。

第一种说明开辟地点并建立访问新开辟地点的名字。后两种说明对已开辟过的地点建立访问它们的新名字。

新地点除了通过地点说明建立外，还可以通过内部子程序**GETSTACK**建立，该子程序将为新开辟的地点提供一个关联值（见下面）。

一个地点可以是**可关联的**。这意味着存在一个该地点的**关联值**。对可关联的地点施行**关联操作**，即可得到关联值。对关联值施行**非关联化**，即可得到所关联的地点。CHILL要求某些地点总是可关联的，但其它地点是否可关联要由实现来确定。可关联性应该是地点的一个可静态确定的性质。

地点可以是**只读**的，这意味着只能通过访问它得到一个值，而不能存放一个新的值进去（除了赋初值）。

地点可以是**组合的**，这意味着拥有可以分别访问的子地点。子地点不一定是可关联的。如果一个地点至少含有一个**只读**的子地点，则称该地点具有**只读性**。提供子地点（或子值）的访问方法分别为：对于字符串和数组是**求子串**、**求元素**和**求片**，对于结构是**选择**。

每个地点有一个模式。若模式是动态的，该地点称为动态模式地点。（注意，动态一词只是对模式而言。尽管运行时地点发生变化，并不意味着它是动态的。地点是动态的，只是说它的性质不能完全静态地确定。）

地点的下列性质虽然可以静态地确定，但不是模式的一部分：

可关联性：对某个地点是否存在一个关联值；

存储类：地点是否被静态地分配；

区域性：该地点是否在某个区域内被说明。

1.5 值及其操作

值是在其上面定义了特定操作的基本对象。一个值是一个(CHILL)**有定义的值**,或一个(在CHILL意义上)**未定义的值**。在指明的上下文中使用未定义的值,将导致出现(在CHILL意义上)无定义情况,此时程序被认为是错误的。

CHILL允许在需要值的上下文处使用地点。在这种情况下,对地点作访问就可取得其中包含的值。

每个值有一个**类**。除了类以外还有模式的值称为**强值**。在这种情况下,这个值总是该模式所定义的值之一。类用于相容性检验,而模式用于描述值的性质。某些上下文要求这些性质是已知的,这时就要求强值。

一个值可以是**直接量**,在此情况下它标志一个与实现无关的,在编译时已知的离散值。值可以是**常量**,在此情况下它永远提供同一个值,即它只要计算一次。**直接量**和**常量**值都假定是在运行之前就计算好了的,它们不可能产生运行时异常。值可以是**区域性的**,在此情况下它能以某种方式与区域地点相关联。值可以是**组合的**,即包含子值。

异名定义语句建立新名字以标志常量值。

1.6 动作

动作构成CHILL程序的算法部分。

赋值动作把(计算好的)值存入一个或多个地点中。**过程调用**调用一个过程,**内部子程序调用**调用一个内部子程序(内部子程序是这样的一个过程,其定义不用CHILL书写,它有较为通用的参数和结果返回机制)。为了从过程调用返回及/或建立过程调用的结果,要使用**结果**和**返回动作**。

为了控制顺序动作流,CHILL提供了下列控制动作流:

条件动作 用于两叉分支;

情况动作 用于多叉分支。分支的选择可以多个值为基础,类似于判定表;

循环动作 用于叠代或加括号;

出口动作 用于以结构化方式中离开加括号内的动作;

引发动作 用于引发一个特定的异常;

转向动作 用于无条件地转到加了标号的程序点。

动作和数据语句可以组合起来以构成一个模块或分程序,后者构成一个(复合)动作。

为了控制并发动作流,CHILL提供了**开动**、**停止**、**延迟**、**继续**、**发送**、**延迟情况**和**接收情况**等动作,或对**接收表达式**进行计值。

1.7 程序结构

程序结构语句有**分程序**、**模块**、**过程**、**进程**和**区域**。程序结构语句提供控制地点的**生存期**和名字的**可见性**的手段。

地点的生存期是指地点在程序内存在的时间。地点可以(在地点说明中)**显式地说明**,或(通过调用内部子程序**GETSTACK**)**生成**,也可以作为某些语言成分的使用结果而**隐式地说明**或**生成**。

如果一个名字可在程序的某一点上使用,则称此名字在该点是**可见的**。名字的**作用域**包含所有使它可见的点,即该名字在那里可以识别它所标志的对象。

分程序既确定名字的可见性,又确定地点的生存期。

模块用于限制名字的可见性,以防止它们被非法使用。利用**可见性语句**可以控制名字在程序的不同部分的可见性。

过程是一个(可能是参数化的)子程序,它可在程序的不同位置被召用(调用),可以送返一个值(**值过程**)或一个地点(**地点过程**),或不提供结果。在后一种情况下,该过程只能在过程调用动作中被调用。

进程和**区域**提供用于实现并发执行结构的手段。

完整的CHILL**程序**是一系列模块或区域,并认为由一个(虚拟的)进程定义所包含。这个最外层的进程

被系统开动，程序就在系统的控制下执行。

1.8 并发执行

CHILL允许各程序单位**并发执行**。**进程**是并发执行的单位。**开动动作**创建它所指定的**进程定义**的一个新进程。此进程即被认为是与开动它的进程并发执行。CHILL允许具有相同或不同定义的一个或多个进程在同一时刻活动。执行**停止动作**导致一个进程终止执行。

每个进程总是处于**活动**或**延迟**这两种**状态**之一。从活动状态转为延迟状态称为进程**延迟**，从延迟状态转为活动状态称为进程**重新活化**。对事件执行延迟动作，对缓冲区或信号执行接收动作，或对缓冲区执行发送动作，都能使执行这些动作的进程被延迟。对事件执行继续动作，对缓冲区或信号执行发送动作，或对缓冲区执行接收动作，都能使被延迟的进程重新活动起来。

缓冲区和**事件**都是地点，它们的使用是有限制的。**发送、接收**和**接收情况**等操作是定义在缓冲区上的，**延迟、延迟情况**和**继续**操作等是定义在事件上的。缓冲区是进程之间同步和传递信息的手段。事件只用于同步。**信号**定义在**信号定义语句**中。它们标志进程之间传递的组合和非组合的值系列、**发送动作**和**接收情况动作**用于值系列的通信和同步。

区域是一种特殊的模块，它用于实现对多个进程共享的数据结构的互斥访问。

1.9 一般语义性质

CHILL的（非上下文无关的）语义条件是模式和类的相容性条件（**模式检验**）及可见性条件（**作用域检验**）。模式检验规则确定名字的使用方式，而作用域检验规则确定名字可以在何处使用。

模式检验规则用模式之间、类之间以及模式和类之间的相容性要求来描述。模式和类之间以及类和类之间的相容性要求用模式之间的等价关系来定义。如果涉及动态模式，则模式检验部分是动态的。

作用域规则通过程序结构和显式的可见性语句确定名字的可见性。显式的可见性语句确定所提到的名字的作用域，也确定这些名字可能**隐含的名字**的作用域。

在程序中引进的名字有一个被定义或说明的地方。这个地方称为该名字的**定义性出现**。而名字被使用的地方称为该名字的**应用性出现**。**名字约束**规则把名字的唯一的定义性出现和它的每一个应用性出现结合起来。

1.10 异常处理

CHILL的动态语义条件是那些（非上下文无关的）一般不能静态地确定的条件。(在运行时是否产生代码以检验动态条件，要留待实现来决定。)违反动态语义条件将引起运行时**异常**。

异常也可以由执行**引发动作**而被引发，还可以由于执行**断言动作**而有条件地被引发。如果该异常是可以指明处理程序的(即它有一个名字)，并且指明了处理程序则当该异常在给定的程序点上出现时，控制就转向相应的异常处理程序。至于是否已经在某一点上为某一异常指明了异常处理程序，是可以静态地确定的。如果没有显式说明的异常处理程序，控制可以转向一个由实现定义的异常处理程序。

大多数异常都有名字。这个名字是一个CHILL定义的异常名字，即实现定义的异常名字，或程序定义的异常名字。注意，当为CHILL定义的异常名字指明一个处理程序时，应该检验有关的动态条件。

1.11 实现任选

CHILL允许有**实现定义的整数模式**、**实现定义的内部子程序**、**实现定义的进程定义**和**实现定义的异常处理程序**。

实现定义的整数模式必须用实现定义的模式名字来标志。这个名字被认为是没有在CHILL语句中说明过的新模式定义语句所定义的。在CHILL语法和语义规则中，允许把现有的CHILL定义的算术操作推广到实现定义的整数模式中去。实现定义的整数模式的例子是**长整数**和**短整数**。

内部子程序是一个未用CHILL语句定义过的过程，它比CHILL过程有更一般的参数传递和结果返回规则。

内部进程名字是一个不用CHILL语句定义的进程名字。CHILL进程可以和实现定义的进程协同运行或开动这些进程。

由实现定义的异常处理程序是附属于虚拟的最外层的进程定义的处理程序。如果在出现异常后转到该处理程序处，则此时应采取什么动作将由实现来决定。

2.0 预备知识

2.1 元语言

CHILL的描述由两部分组成：

- 上下文无关语法的描述；
- 语义条件的描述。

2.1.1 上下文无关语法的描述

上下文无关语法用扩充的巴科斯——瑙尔范式描述。语法范畴是用一个或多个异体汉字词组(或英文词组)组成并用尖括号将其括起来（〈和〉），称之为非终结符号。对于每个非终结符号，都有相应的语法部分给出其产生规则。非终结符号的产生规则是这样组成的：在符号::=的左边是这个非终结符号，在符号::=的右边是由非终结产生式和/或终结产生式组成的一个或多个结构。这些结构之间用竖线(|)分开，它们代表这个非终结符号的可供选用的产生式。

有时，非终结符号包括一个下划线的部分。这个下划线的部分并不构成上下文无关描述的一部分，而是定义了一个语义子范畴（参见2.1.2节）。

语法元素可以用花括号（{和}）括在一起。花括号组的重复用星号(*)或加号（+）表示。星号表示此组合是任选的，也可以重复任意多次；加号表示此组合必须出现，并可重复任意多次。例如，[A]*表示由任意多个A，包括零个A，组成的序列，而{A}+表示由至少一个A组成的任意序列。如果语法元素用方括号([和])括起来，则此组合是任选的。

严格语法和**导出语法**是有区别的。严格语法的语义条件是直接给出的。导出语法可以看作是严格语法的扩充。导出语法的语义用相应的严格语法间接解释。

注意，在本文件中，上下文无关语法的描述方式是根据语义描述的需要选定的，并不是为了适应任何特定的语法分解算法的需要（例如，为了表达清楚，引进了一些上下文无关的二义性）。

2.1.2 语义描述

每个语法范畴（非终结符号）的语义描述在**语义**、**静态性质**、**动态性质**、**静态条件**和**动态条件**等部分给出。

语义部分描述由语法范畴表示的概念（即它们的意义和行为）。

静态性质部分定义该语法范畴的可静态地确定的语义性质。在该语法范畴在相应的章节使用时，这些性质可用来描述其静态的和/或动态的条件。

动态性质部分视需要而设立，定义此语法范畴的只能是动态确定的性质。

静态条件部分描述上下文有关的并可静态地检验的条件，当用到该语法范畴时，这些条件必须被满足。在语法中，某些静态条件是通过非终结符号中的下划线部分来表达的(参见2.1.1节)。这种使用方式要求该非终结符是一个特定的语义子范畴。例如，〈布尔表达式〉在上下文无关的意义下恒等于〈表达式〉，但是在语义上要求该表达式属于布尔类。下划线部分有时在文本中用作形容词以限定非终结符。例如，“此表达式是**常量**”这句话相当于“此表达式是常量表达式”。

动态条件部分描述在程序执行时应该满足的上下文有关条件。在某些情况下，当且仅当不涉及动态模式时，条件是静态的。在这种情况下，该条件在**静态条件**部分被提到，而在**动态条件**部分被引用。

在语义描述中，非终结符号用不带尖括号的异体写出，以此来标示语法对象。

2.1.3 例子

对大多数语法章节来说，都有一个例子部分，其中给出所定义的语法范畴的一个或几个例子。这些例子是从附录D中作为例题的一组程序中抽出来的。索引指出每个例子是通过哪一个语法规则产生的，以及它是从哪一个例题中抽取出来的。

例如，6.20 (d+ 5)/5 (1.2) 表示例子是一个终结字符串 (d+5)/5，它由相应语法章节的规则 (1.2) 产生，取自程序例题 6 的第20行。

2.1.4 元语言的约束规则

有时语义描述提到CHILL专用的名字(见附录C)。这些专用名字在使用中总是带有它的CHILL意义，因此不受实际CHILL程序约束规则的影响。

2.2 词汇表

程序用CCITT第5号字母表，建议书V.3（见附录A1）表示。但每个CHILL程序都可以用一个最小字符集表示，它是CCITT第5号字母表基本码的一个子集（见附录A2）。

CHILL的词法元素是：

- 特殊符号。
- 名字。
- 直接量。

特殊符号在附录B中给出。

名字根据下列语法构成：

语法：

〈名字〉::= (1)

〈字母〉{〈字母〉|〈数字〉|_}* (1.1)

下划线符号{_}是名字的一部分。即，名字LIFE-TIME不同于名字LIFETIME。在有小写字母可供使用的情况下，这些字母也可用于名字中。小写字母和大写字母是不同的，例如Status和status是两个不同的名字。

本语言有一些具有预先确定意义的专用名字，见附录C。它们中有些是保留的，即，除非用释放命令显式地释放，它们是不能派作其它用途的。

在大、小写字母都可以使用的情况下，专用名字可以或者全用大写字母表示，或者全用小写字母表示。保留名字只在选定的表示形式中保留（例如，若选定小写形式，则row是保留的，而ROW则不保留）。

2.3 空格的使用

空格可用来作为程序中词法元素的分隔。词法元素在第一个不能成为词法元素一部分的字符处结束。例如，IFBTHEN将被认为是一个名字而不是一个动作IF B THEN的开始。//*将被看作是连接符{//}后面跟一个星号{*}，而不是一个除号{/}后面跟一个注释起始括号{/*}。连续的空格和单个空格具有相同的分隔效果。

2.4 注释

语法：

〈注释〉::= (1)

/* 〈字符串〉*/ (1.1)

〈字符串〉::= (2)

{〈字符〉}* (2.1)

语义 注释向阅读者传递有关程序的信息。它对程序的语义没有影响。

静态性质 只要是允许使用空格作分隔符的地方，都可以插入注释。

静态条件 字符串不得包含特殊序列{*/}。

例子:

4.1 /*from collected algorithm from CACM nr. 93 */ (1.1)

2.5 格式作用符

在CHILL上下文无关语法描述中没有提到CCITT第5号字母表（从位置FE_0到FE_5）的格式作用符BS（退一格），CR（回车），FF（换页），HT（水平制表），LF（换行）和VT（垂直制表）。但是，在实现时可以在CHILL程序中使用这些格式控制符。当使用时，它们有和空格一样的分隔作用，但不能在词法元素内部使用。

2.6 编译命令

语法:

〈命令子句〉::= (1)

〈〉〈命令〉{,〈命令〉}*〔〈〉〕 (1.1)

〈命令〉::= (2)

〈CHILL命令〉 (2.1)

|〈实现命令〉 (2.2)

〈CHILL命令〉::= (3)

〈释放命令〉 (3.1)

〈释放命令〉::= (4)

FREE(〈保留名字表〉) (4.1)

〈名字表〉::= (5)

〈名字〉{,〈名字〉}* (5.1)

语义: 命令子句向编译程序提供信息。除释放命令外，该信息由实现定义的格式来说明。

实现命令不得影响程序的语义。即，当且仅当不带实现命令的程序在CHILL意义上是正确时，加上实现命令的同一程序也是正确的。

释放命令用于编译单元，它将释放在保留名字表中指明的保留名字，使它们可以在该编译单元中被重新定义。

静态性质 命令子句可以插入所有允许空格出现的地方。它与空格有同样的分隔作用。在命令子句中使用的名字服从实现定义的名字约束规则，该规则不影响CHILL的名字约束规则（参见9.2.8节）。

静态条件 任选的命令结束符号{〈〉}只当它正好位于一个分号之前时方可被省去（即命令子句以第一个〈〉或分号结束。但分号不属于命令子句。因此，命令既不能包含〈〉符号，也不能包含分号，除非它位于圆括号中间，参见下面）。若圆括号在实现命令中出现，它们必须正确地配对，并且，如果一个分号或命令结束符号出现于圆括号之内，则它们并不终止该命令。

例子:

15.1 〈〉FREE(STEP) (1.1)

15.1 FREE(STEP) (4.1)