ISSN1000-9825 Journal of Software 软件学报 2000,11(1):116~121

区间逻辑的一个辅助证明工具*

胡成军 王戟 陈火旺

(国防科学技术大学计算机学院 长沙 410073)

摘要 DC/P(duration calculus prover)是一族实时区间逻辑的辅助定理证明工具.它采用Gentzen风格相继式演算作为基本证明系统,并结合项重写、自动判定算法等技术以提高证明的自动化程序.该文介绍了DC/P的语义编码方法、采用的相继式证明系统及实现技术,并给出了应用实例.

关键词 邻域逻辑,区间时序逻辑,均值演算,时段演算,相继式演算,定理证明.

中图法分类号 TP301

作为一种实时逻辑,区间逻辑因其表达能力强而被大量应用于硬件电路、实时系统等领域[1,2].区间逻辑区别于点式逻辑(point based temporal logic)的根本性质是公式语义解释在区间上,而不是在时间点上.对于某些性质,例如,"A 在任一长于 l 的观察区间上总为真"等,使用区间逻辑来描述、推导更方便、更自然.

本文介绍我们开发的时段演算定理证明器(duration calculus prover,简称DC/P)(DC/P版本1.0可从http://www.iist.unu.edu/~hcj下载,需要者也可与作者联系).作为区间逻辑的一个辅助证明工具,DC/P的目的是将推导过程机械化、(半)自动化,使人们从枯燥易犯错的手工推导中解脱出来.DC/P实际上是支持包括邻域逻辑[3](neighbourhood logic,简称NL)、区间时序逻辑[4,5](interval temporal logic,简称ITL)、均值演算[6](mean calculus,简称MC)和时段演算[7](duration calculus,简称DC)在内的一族实时区间逻辑的辅助定理证明工具.之前,Skakkebæk的PC/DC[8]是我们所知的唯一的时段演算辅助证明工具.PC/DC是一个出色的工具,其缺点是仅支持DC且很难被用户扩充(PC/DC采用了所谓的"Parser/Unparser"技术,这一技术因过于依赖实现平台的特定功能而难于维护).DC/P吸收了PC/DC的一些优点,并在此基础上作了一些改进.DC/P是交互式的,它采用Gentzen风格相继式演算(sequent calculus)作为基本证明系统,同时也结合了如重写、自动判定算法等技术,用户只需在高层作一些决策,其他大量琐碎的细节则由DC/P自动完成.

DC/P的实现平台是SRI公司著名的原型验证系统PVS[9].在实现DC/P时,我们采用语义编码技术.采用语义编码技术而不是从头构造一个全新定理证明工具的主要原因有两方面,一方面我们可以较快地得到一个可靠性有保障的原型系统,另一方面我们可以将注意力更多地放在逻辑的证明系统和相关自动化技术上.DC/P的实现方法对其他逻辑的证明工具的实现均有参考意义.

1 语义编码

我们采用的语义编码是在文献[10]的基础上作了较大改动后得到的.其中,域、状态、项及公式等编码如下.

我们采用PVS的内置类型real来表示时间域.区间类型则用时间点的有序点对来表示:

```
Time:TYPE=real;
Interval:TYPE={(b:Time,e:Time)|e>=b};
```

* 本文研究得到国家自然科学基金(No.69603010,69873045)、国家863高科技项目基金(No.863-306-ZT06-04-1)和UNU/IIST(International Institute for Software Technology, United Nations University)项目基金资助.作者胡成军,1972年生,博士,讲师,主要研究领域为形式化方法.王戟,1969年生,博士,副教授,主要研究领域为软件工程.陈火旺,1935年生,教授,博士生导师,中国工程院院士,主要研究领域为软件工程,人工智能.

本文通讯联系人:胡成军,青岛 266071,海军潜艇学院指战教研室

本文1998-10-22收到原稿,1999-01-22收到修改稿

状态通常表示为从时间域到布尔域中的函数.然而,由于状态变元(及表达式)仅出现在积分算子内,通常假定状态变元具有有穷可变性(finite variability)以使积分函数是良定的.这一约束用 PVS 的谓词子类型描述如下:

```
State:TYPE={f:[Time->T]|Finite_Variability(f)};
boolstate:TYPE=State[bool];
```

这里,T 为一类型参数.*Finite_Variability*(f)是一递归定义的高阶谓词,其含义是,在任一区间上,总存在一自然数 n 使得函数 f 恰好有 n 个不连续点.

我们可以在状态上施加各种"布尔型"操作以得到状态表达式.状态间的联接算子可由"提升"的方法定义.例如令 $F1$,$F2$ 为 boolstate 类型,t 为一个 Time 类型变元,则"蕴含"算子可定义为

```
=>(F1,F2):boolstate=(LAMBDA t:F1(t) IMPLIES F2(t));
```

区间逻辑的项由全局变元、时序变元、函数及常数构成.这里,我们把状态表达式的积分(或均值)视为特殊的时序变元,并借助类型信息来区分全局变元与时序变元.

```
GV:TYPE=real;                    % global variables
TV:TYPE=[Interval->real];        % temporal variables
Term:TYPE=TV;                    % an alias of TV,terms
l:Term=(LAMBDA(i: interval): proj_2(i)-proj_1(i));
```

当全局变元或实数型常数用作项时,需使用类型强制(type coercion).令 i 为 Interval 类型,类型转换函数定义如下:

```
n(t:T):[Interval->T]=LAMBDA i:t;  CONVERSION n;
```

这样定义的类型转换函数 n 是多态的,它可以将任一类型 T 转换为另一类型[Interval->T].如从实数到项、从布尔型到公式等.由于 PVS 的类型检查系统自动尝试调用用户定义的转换函数以获得类型匹配,因此,在规范中,我们仍可以用常规方式书写各种表达式,如 l+0,gx*5 等.

采用类似方法,实数域上通常的四则运算可以提升到项上.如项上的乘法运算定义为

```
*(t1,t2):Term=(LAMBDA i:t1(i)*t2(i));
```

在 DC/P 中,我们没有把 Reiman 的积分理论编码到元逻辑中,而是将积分算子解释为未定义函数,并引入一组公理来刻画其语义.文献[11]已经证明该公理化是完备的.积分算子及均值函数定义如下:

```
intergral(s):[Interval->nonneg_real];
dur(s):Term=intergral(s);
MV(s):[Interval->nonneg_real]=(LAMBDA i:IF proj_1(i)=proj_2(i)
        THEN IF s(proj_2(i)) THEN 1 ELSE 0 ENDIF
        ELSE intergral(s)(i)/(proj_2(i)-proj_1(i)) ENDIF;
```

区间逻辑的公式是从区间到布尔域的函数 Form:TYPE=[Interval->bool].同样地,通过提升的方法,我们可以得到更多的关系算子、命题联接词及量词.例如:

```
>=(t1,t2):Form=(LAMBDA i:t1(i)>=t2(i));
\/(A,B):Form=(LAMBDA i:A(i) OR B(i));
every(L:[T->Form]):Form=(LAMBDA i:(FORALL gx: L(gx)(i))).
```

而 NL 的模态词$\Diamond_l$及 ITL 的"chop"模态词$\frown$,$\Diamond$和$\Box$定义如下:

```
dl(A):Form=(LAMBDA i:(EXISTS (delta:Time):A(proj_1(i)-delta,proj_1(i))));
^(A,B):Form=(LAMBDA i:(EXISTS (m:Time):(proj_1(i)<=m) AND (m<=proj_2(i)) AND
        A(proj_1(i),m) AND B(m,proj_2(i))));
()A:Form=tt^A^tt;
[]A:Form=not(()(not A)).
```

区间逻辑的公式语义可以按统一的形式编码到元逻辑中.直观上讲,公式 A 在区间 i 处为真当且仅当 $A(i)$

等价于永真式 *true*;公式 A 为真当且仅当 $A(i)$对于所有区间 i 都为真.对此可以编码如下:

```
valid(A):bool=FORALL i: A(i);
|-(A):bool=valid(A).
```

2　Gentzen 风格证明系统

在完成语义编码后,原则上可以通过展开所有算子的定义而在元逻辑层次来证明区间逻辑的定理.但这样一来,一方面证明工具的用户必须熟悉编码细节;另一方面这种证明没有反映人们在区间逻辑概念层次的思维,最终得到的证明将难以理解.为克服上述缺点,我们开发了一个 Gentzen 风格的证明系统.

2.1　推导规则

DC/P 的证明系统包含了一阶逻辑 Gentzen 相继式演算的所有公理与规则,其余的公理、规则基本上是 Dutertre 的公理化系统 S′[5]以及文献[12]中 DC 公理化系统的相继式风格的对应.限于篇幅,本节仅列出与"chop"算子、特殊符号 ϵ 等相关的部分规则及 DC 的归纳规则.

$$\text{ILR1a}:\frac{\Gamma,\beta\frown(\psi\wedge\neg\varphi)\to\Delta}{\Gamma,\beta\frown\psi\to\beta\frown\varphi,\Delta}\qquad \text{ILR2a}:\frac{\Gamma\to(\epsilon=x)\frown\neg\beta,\Delta}{\Gamma,(\epsilon=x)\frown\beta\to\Delta}\qquad \text{ILR3a}:\frac{\Gamma,\psi\frown\varphi\to\Delta\quad \Gamma\to\Box(\beta\Rightarrow\psi),\Delta}{\Gamma,\beta\frown\varphi\to\Delta}$$

$$\text{ILR4a}:\frac{\Gamma\to\neg\beta\frown\psi,\Delta\quad \Gamma\to\Box\beta,\Delta}{\Gamma\to\Delta}\qquad \text{DCR1a}:\frac{\Gamma\to R(\lceil\rceil),\Delta\quad \Gamma,R(X)\to R(X\vee(X\frown(\lceil S\rceil)\vee(X\frown\lceil\neg S\rceil)),\Delta}{\Gamma\to R(tt),\Delta}$$

除基本规则外,DC/P 中还包含许多功能强大的派生规则.这些规则使得证明时能够利用较大的证明步骤,从而使证明更简洁.如在证明含◇,□算子的定理时,应用模态逻辑 S4 的如下规则往往更为方便.

$$\Box\to:\frac{\Box\Gamma,\beta\to\Diamond\Delta}{\Box\Gamma,\Diamond\beta\to\Diamond\Delta}\qquad \to\Diamond:\frac{\Gamma\to\beta,\Delta}{\Gamma\to\Diamond\beta,\Delta}\qquad \Box\to:\frac{\Gamma,\beta\to\Delta}{\Gamma,\Box\beta\to\Delta}\qquad \to\Box:\frac{\Box\Gamma\to\beta,\Diamond\Delta}{\Box\Gamma\to\Box\beta,\Diamond\Delta}$$

这里,□Γ(或◇Δ)表示有穷公式集,其中每一公式或者是刚性(rigid)的,即不包含任意时序变元、ϵ 和模态词;或者是□(或◇)公式,即形如$\lceil\Box\rceil$(...)(或◇(...))的公式.

下面的定理[13]给出了我们所采用的相继式演算的可靠性及相对完备性的结论.

定理 1. 若公式 β 是 DC/P 的定理,则 β 在标准的区间语义模型下永真.

定理 2. 若公式 β 存在一个 Hilbert 风格的证明,则 β 是 DC/P 的定理.

2.2　规则编码

首先需将相继式及规则编码到元逻辑中.相继式 $\Gamma\to\Delta$ 在区间 i 处为真的含义是:存在公式 $A\in\Gamma$,$A(i)$为假,或存在 $B\in\Delta$,$B(i)$为真.若相继式在任意区间上都为真,则称相继式永真.令 *Gamma*,*Delta* 为 Form 类型,相继式编码如下:

```
seq(Gamma,Delta):bool=|-(Gamma=>Delta)
```

公式的有穷集通过枚举出成员的方法表示.引入如下算子构造有穷集:

```
emptyconsequent:Form=ff;  emptyantecedent:Form=tt;
*(Gamma,X):Form=Gamma/\X;  +(Delta,X):Form=Delta\/X;
```

这里,算子+,*用于连接有穷集中的各个成员,不同于区间逻辑中的加乘运算符.例如,相继式{A,B}→{C,D,E}在 DC/P 内部实际表示为元逻辑公式 seq(enptyantdcedent * A * B,enptyconsequent | C | D | E).由于我们修改了 PVS 的接口,上述编码对用户而言是透明的.给出上述算子的定义后,DC/P 的推导规则及公理就可以编码为元逻辑的公式.若规则或公理是可靠的,则我们还应该可以证明它们是元逻辑的定理.以"chop"算子左单调性规则(即 ILR3a)为例,它可以编码如下:

```
ILR3a:THEOREM seq(Gamma * (A ^ B),Delta) AND seq(Gamma,Delta+[](C=>A))
          IMPLIES seq(Gamma * (C ^ B),Delta)
```

2.3　证明命令

DC/P 中直接与用户交互的是一组证明命令.证明命令的用途是选择相应的规则作用于当前待证目标,得

到一组简化的子目标或结束证明.证明命令通常对应于1条或多条推导规则,可视为一些"智能"的宏命令.换言之,所有证明命令最终都可以展开为一组基本的PVS证明命令.这种实现方式确保了当引入新命令时不会影响系统的可靠性.

DC/P的基本证明命令最终都是通过引入编码后的公理或规则实例实现的.以"chop"算子左单调命令MONO-CHOP为例,DC/P首先读入用户输入,并将其与编码后的规则ILR3a的前提进行匹配,若失败,则显示错误信息并退出;否则找到$Gamma$,$Delta$,A,B,C的代入项,引入实例化后的规则并进行化简,从而将当前相继式规约为规则ILR3a的结论.DC/P的大部分证明命令都对应于多条规则,如命令NFLATTEN,NSPLIT分别对应于命题逻辑中产生单一子目标、产生多子目标的规则.实现这些命令时需要对多条规则一一匹配.

简单命令可用PVS策略语言所提供的顺序、条件和重复等结构复合而成.如命令NPROP可用于证明所有命题逻辑定理,它是通过在所有待证目标上反复尝试应用命令NFLATTEN和NSPLIT实现的.NPROP定义为

(TRY (NFLATTEN) (NPROP) (TRY (NSPLIT) (NPROP) (SKIP)).

这里,TRY相当于IF...THEN...ELSE...结构.上式将首先调用命令NFLATTEN,若有新子目标,则递归调用自身,否则调用下一个TRY命令.

复杂证明命令大多数通过如下步骤实现:首先扫描分析当前待证目标,判定需应用的规则;然后根据情况可能需要对待证目标作一些变换,并将推导规则所表示的定理实例化;最后简化目标.考虑如下简单定理:

$$\vdash \int Leak \leqslant x \frown \int Leak \leqslant y \Rightarrow \int Leak \leqslant x+y.$$

在手写证明时,上述定理的证明过程往往被忽略,而错误容易出现在这种被忽略的地方.严格的形式化验证过程不应该忽略任何细节,而人们通常会希望类似于上面的"显而易见"的定理能由辅助证明工具自动完成.在DC/P中,如果仅使用基本规则,一个可能的推导类似于如下手写证明:

$$\begin{aligned}
&\int Leak \leqslant x \frown \int Leak \leqslant y \\
\Rightarrow &(\int Leak = x' \wedge \int Leak \leqslant x) \frown (\int Leak = y' \wedge \int Leak \leqslant y) \ \text{(Exist-intro)} \\
\Rightarrow &(\int Leak = x' \wedge x' \leqslant x) \frown (\int Leak = y' \wedge y' \leqslant y) \ \text{(Chop-mono)} \\
\Rightarrow &(\int Leak = x' \frown \int Leak = y') \wedge x' \leqslant x \wedge y' \leqslant y \ \text{(Rigid-out)} \\
\Rightarrow &\int Leak = x' + y' \wedge x' \leqslant x \wedge y' \leqslant y \ \text{(Dc-axiom5)} \\
\Rightarrow &\int Leak \leqslant x + y \ \text{(Nassert)}
\end{aligned}$$

这里,Exist-intro等是DC/P的证明规则(命令).规则Exist-intro的功能是为时序变元引入skolem常数x',y'.Chop-mono是"chop"算子的单调性规则,即ILR3.规则Rigid-out将所有刚性公式移出到chop算子的辖域外.最后,Nassert是一条复杂的命令,它首先对当前目标作一些变换,然后调用实线性算术的自动判定过程.上述的证明过程也可以由DC/P命令Dur-chop自动完成.Dur-chop是一条复杂的DC/P证明命令,它结合了一系列在上述手写风格证明中出现过的子命令,完成大量琐碎的工作.首先,Dur chop扫描当前相继式,寻找一个形如$p(\int s) \frown q(\int s)$的前提相继式公式,这里,$p(\int s)$和$q(\int s)$是关于$\int s$(若将其看做变元)的实线性算术公式.然后,命令Exist-intro被执行以引入skolem常数,通过命令Chop-mono和Rigid-out,相继式被转换成最终可直接应用DC公理5的形式,最后通过调用实线性算术的自动判定算法以证明当前目标.在DC/P中,还有不少类似的复杂命令.正是由于这些命令的存在,使得大多数琐碎的证明步骤得以简化.

3 应用实例

考虑一个可能会发生泄漏*的煤气燃烧炉[7],我们希望它满足安全性质:在任何大于1分钟的观察区间内,漏气的时间不超过整个观察区间的5%.该需求规范可以用时段演算公式描述如下:$Req \triangleq \ell \geqslant 60 \Rightarrow 20\int Leak \leqslant \ell$.为了满足这一需求,设计人员需要作出一些决策.例如,根据经验,如果煤气炉控制系统满足:(1) 任何一次漏气事件都可以在1秒内检测到;(2) 发现一次漏气后,煤气炉的阀门将被关闭,并至少等待30s才会重新打开,已知这样设计的煤气炉是满足上述安全性的.上述设计策略可以用DC公式描述如下:

$$Des1 \triangleq \Box(\lceil Leak \rceil \Rightarrow \ell \leqslant 1)$$

$$Des2 \triangleq \Box(\lceil Leak \rceil \frown \neg Leak \rceil \frown \lceil Leak \rceil \Rightarrow \ell \geqslant 30)$$

为了验证设计符合规范,我们需证明 $Des1 \wedge Des2 \Rightarrow Req$ 是DC的定理.借助于DC/P,这一形式化验证过程被机械化了.当证明过程成功地完成时,我们有理由确信设计的正确性.上述公式在DC/P中描述如下:

```
Req: Form=(1>=60)=>(20*dur(Leak)<=1);
Des_1: Form=[](([|Leak|]=>1<=1);
Des_2: Form=[]((([|Leak|]^[|not(Leak)|]^[|Leak|]=>1>=30);
Gasburner: THEOREM|-(Des_1/\Des_2=>Req);
```

上一节中给出的简单定理是煤气炉案例的一个引理.下面我们给出该引理在DC/P中的证明过程,以给读者一些印象.

```
|DC ------------------------------------------------------------
(1)    (dur(s! 1)<= n(x1! 1)) ^ (dur(s! 1) <= n(x2! 1))
       =>(dur(s! 1) <= n(x1! 1 + x2! 1))
Rule? (APPLY (THEN (NFLATTEN) (DUR-CHOP)))
Apply composite rules,... Q.E.D.
```

4 结束语

本文描述了实时区间逻辑的一个辅助定理证明工具DC/P.DC/P的主要优点有以下几点:

- 采用相继式演算作为基本证明系统,一阶逻辑部分的证明在很大程度上被自动化;
- 由于重写技术、实线性算术判定算法的应用,很多琐碎的证明被简化;
- 集成了一个时段演算子集的自动判定过程;
- 与PC/DC相比,DC/P很容易被用户扩充,如增加新的逻辑算子、证明命令的定制等.

目前,我们正在尝试在DC/P中嵌入一条带标记的相继式演算.初步试验表明,这一演算能利用语义知识来指导证明搜索过程,从而进一步提高证明的自动化程度.

致谢 Dimitar P. Gelev博士细心校读本文英文稿初稿,徐启文博士、Dang Van Hung博士、周巢尘教授及DeTfoRs组的其余组员在多次讨论中提出许多有益建议,我们在此表示感谢!

参考文献

1 Hansen M R, Zhou Chao-chen, Stauntrup J. A real-time duration semantics for circuits. In: TAU 1992 ACM/SIGDA Workshop on Timing Issues in the Specification and Synthesis of Digital Systems. Princeton, NJ, 1992

2 Ravn A P, Rischel H, Hansen K M. Specifying and verifying requirements of real-time systems. IEEE Transactions on

* 特指煤气阀门处于开的位置,但炉中无火苗的状态.

Software Engineering, 1993,19(1):41～55

3 Zhou Chao-chen, Hansen M R. An adequate first order interval logic. In: de Roever W P, Langmaack H, Pnueli A eds. International Symposium on Compositionality——the Significant Difference (COMPOS'97). Germany: Malente/Holstein, 1997

4 Moszkowski B. A temporal logic for multilevel reasoning about hardware. IEEE Computer, 1985,18(2):10～19

5 Dutertre Bruno. On first order interval temporal logic. Technical Report, No. CSD-TR 94 3, Department of Computer Science, Royal Holloway University of London, 1995

6 Zhou Chao-chen, Li Xiao-shan. A mean value calculus of durations. In: Roscoe A W ed. A Classical Mind: Essays in Honor of C. A. R. Hoare. Englewood Cliffs, NJ: Prentice Hall International, 1994. 431～451

7 Zhou Chao-chen, Hoare C A R, Ravn A P. A calculus of durations. Information Processing Letters, 1991,40(5):269～276

8 Skakkebæk J U, Shankar N. Towards a duration calculus proof assistant in PVS. In: Langmack H, de Roever W P, Vytopil J eds. Formal Techniques in Real-Time and Fault-Tolerant Systems. LNCS 863, Mook, the Netherlands: Springer-Verlag, 1994. 660～679

9 Shankar N, Owre S, Rushby J M. The PVS Proof Checker: a Reference Manual (Beta Release). Computer Science Laboratory, Menlo Park, CA: SRI International, 1993

10 Mao Xiao-guang, Xu Qi-wen, Wang Ji. Towards a proof assistant for interval logics. Technical report, International Institute for Software Technology, United Nations University. 1996

11 Hansen M R, Zhou Chao-chen. Semantics and completeness of duration calculus. In: de Roever W P, de Bakker J W, Huizing C eds. Real-Time: Theory in Practice, REX Workshop. LNCS 600, Mook, the Netherlands: Springer-Verlag, 1992. 209～225

12 Hansen M R, Zhou Chao-chen. Duration calculus: logical foundations. Formal Aspects of Computing, 1997,9(3):283～330

13 Hu Cheng-jun. DC/P: a proof assistant for interval logics. Technical Report, International Institute for Software Technology, United Nations University, 1999

A Proof Assistant for Interval Logics

HU Cheng-jun WANG Ji CHEN Huo-wang

(*School of Computer National University of Defense Technology Changsha* 410073)

Abstract DC/P (duration calculus prover) is a proof assistant for a family of interval logics. It adopts the Gentzen-style sequent calculus as its basic proof system. The techniques such as term rewriting and automatic decision procedure are integrated to automate many trivial proof steps. In this paper, the authors briefly describe the semantic encoding approach, and the sequent calculus, as well as the related implementation techniques of the DC/P.

Key words Neighbourhood logic, interval temporal logic, mean calculus, duration calculus, sequent calculus, theorem proving.