具有行为观察的移动界程演算空间逻辑

陈 江,林荣德

(华侨大学 网络信息中心,福建 泉州 362021)

[摘要] 界程逻辑中的并发模态副词是观察进程交互行为的关键因素之一,但引入并发模态副词又会导致模型检测的不可判定性. 针对这一问题,提出了可判定的、描述移动界程演算进程的空间结构和行为性质的应用界程逻辑. 该逻辑定义了空间模态词和行为模态词来直接观察移动进程的空间性质和潜在交互行为性质,并定义了不动点公式来刻画进程的递归性质. 为了证明逻辑公式的指称语义的正确性,引入了性质集的概念并证明两者之间的一致性. 最后给出了使用应用界程逻辑公式来描述一个资源传输系统在时间和空间上的行为性质的实例.

[关键词] 移动界程演算,界程逻辑,空间和行为观察,模型检测

[中图分类号] TP311 [文献标志码] A [文章编号] 1672-1292(2011)04-0070-07

Spatial Logic With Behavioral Observations for Ambient Calculus

Chen Jiang, Lin Rongde

(Computer Network and Information Center, Huaqiao University, Quanzhou 362021, China)

Abstract: A decidable ambient based spatial logic, named Applied Ambient Logic is proposed, which consists of spatial modal connectives and behavioral modal connectives. In traditional ambient logic, the composition adjunct is very expressive, which makes it possible to observe behavioral properties of processes. However, it is proved that model-checking of logics with composition adjunct is not decidable. In Applied Ambient Logic, both spatial modal connectives and behavioral modal connectives are added which can specify spatial and behavioral properties of processes directly, and fix-point modals are also added for properties of recursive processes. A concept of property sets is adopted to describe the features of denotational semantics for logic formulas, and soundness between property-set and denotational semantics is proved. Finally, examples of Applied-Ambient Logic formulas are given which are applied to describe spatial-temporal behavioral properties of a resource transformation system model.

Key words: ambient calculus, ambient based logic, spatial and behavioral observations, model checking

界程逻辑[1](AL逻辑)是用来表达界程演算模型[2]性质的模态逻辑. AL逻辑通过3个最基本空间公式:0,$A \mid B$和$n[A]$来直接描述移动界程演算进程的空间性质,其中钝性公式0描述钝性进程0,并发组合公式$A \mid B$则描述进程是一个并行组合$P \mid Q$且P满足A和Q满足B,而界程公式$n[A]$则描述进程是一个形如$n[P]$的界程且P满足A. 为了描述界程演算的进程的动作行为,AL用空间并发模态副词$\triangleright$和一个最小Next或Eventually的行为模态词,以间接方式来描述进程的动作行为. 空间模态副词$\triangleright$有很强的表达能力[3,4],然而AL逻辑在带有模态副词$\triangleright$的情况下进行模型检测是不可判定的[5-7].

为了同时描述界程演算进程所表达的空间性质和行为性质,本文给出了一种可判定的、具有行为模态描述能力的界程演算空间逻辑——应用界程逻辑(Applied-AL逻辑). 借助于空间模态词和行为模态词来直接观察进程的本地结构和行为,Applied-AL逻辑可以不必包含空间模态副词$\triangleright$,并保持描述进程行为性质和空间性质的能力. 在空间模态部分,该逻辑考虑了界程、空间并行组合、钝性和揭示模态词;在行为模态部分,该逻辑考虑了用$\langle\cdot\rangle$表示进程的Next时态行为,$\langle \mathrm{cap}\ n\rangle$描述进程执行一个移动行为,或通过$\langle n!m\rangle$和$\langle n?m\rangle$分别描述进程向界程$n$发送或接收名字$m$;在基本的逻辑连接词部分,Applied-AL逻辑

收稿日期:2011-04-11.
基金项目:华侨大学科研基金(11BS121).
通讯联系人:陈 江,讲师,研究方向:网络计算、形式化建模等. E-mail:chenj@hqu.edu.cn

考虑了命题操作符、一阶量词和新鲜名量词以及最大不动点的递归操作符.

为了精确给出该逻辑在公平界程演算[8]下的逻辑语义,采用了一个特殊的进程集(称为性质集)来定义逻辑公式的指称语义. 建立了性质集与公式的指称语义之间的一致性,从而为检测 Applied-AL 逻辑公式的满足性,特别是不动点公式的满足性算法提供理论基础.

本文给出了移动界程演算的相关概念及应用界程逻辑公式的语法,引入了性质集来描述逻辑公式指称语义,证明了包含空间模态、行为模态的 Applied-AL 逻辑的指称语义的正确性,并使用 Applied-AL 逻辑公式描述一个资源传输系统模型在不同时间空间上行为性质的实例.

1　相关概念

本文涉及的界程演算模型是来自于公平界程演算[8]的有限控制片断,本节仅简要介绍进程的语法、结构同余和归约语义. 详细内容可参考相关文献[8-10].

令 P 表示所有界程演算进程构成的集合,进程的语法由如下 BNF 表示:

动作前缀 $\mathrm{cap}n \in \{\mathrm{in}\ n, \overline{\mathrm{in}}\ n, \mathrm{out}\ n, \overline{\mathrm{out}}\ n, \mathrm{open}\ n, \overline{\mathrm{open}}\ n \mid n \in N\}$;

通讯前缀 $\pi \in \{n(x), n!x \mid n, x \in N\}$;

进程 $P, Q ::= 0 \mid (\nu n)P \mid P \mid Q \mid n[P] \mid \mathrm{cap}\ n.P \mid \pi.P \mid \mathrm{rec}\ X = P$.

0 是不与其他进程发生任何交互的进程. $(\nu n)P$ 是带名字限制的进程. $P \mid Q$ 是并行组合的进程. $n[P]$ 表示一个名字为 n、内部运行进程 P 的带边界的进程,简称为"界程 n". $\mathrm{cap}\ n.P$ 表示进程带有可以顺序执行的能力前缀 $\mathrm{cap}\ n$,其中 $\mathrm{in}\ n$ 和 $\overline{\mathrm{in}}\ n$、$\mathrm{out}\ n$ 和 $\overline{\mathrm{out}}\ n$,以及 $\mathrm{open}\ n$ 和 $\overline{\mathrm{open}}\ n$ 之间互为动作-协动作关系. $\pi.P$ 表示进程带有一个形如 $n(x)$ 或 $n!x$ 的通讯前缀,其中 $n(x)$ 和 $n!x$ 之间形成了动作-协动作关系. $\mathrm{rec}\ X = P$ 表示子进程 P 被递归执行,有限控制的递归进程需要类型系统来静态约束它的行为,相关类型系统参考相关文献[9,10].

界程演算的静态语义由进程的结构同余关系 $\equiv$ 来表示,它是由表 1 所示规则生成的进程上的最小同余关系. 界程演算的动态语义由进程的归约关系 $\rightarrow$ 来表示,它由表 2 中所列规则所定义. 表 2 中的前 4 个规则(Red In)、(Red Out)、(Red Open)和(Red Comm)表示界程之间执行迁入、迁出和打开等动作,以及进行界程之间通讯都需要交互双方的对等授权和身份确认.

表 1　界程演算进程的结构同余关系 $\equiv$

Table 1　Structural congruence rules of mobile ambients

(Stru Refl) $P \equiv P$	(Stru Cap) $P \equiv Q \Rightarrow \mathrm{cap}\ m.P \equiv \mathrm{cap}\ m.Q$
(Stru Symm) $P \equiv Q \Rightarrow Q \equiv P$	(Stru Oup) $P \equiv Q \Rightarrow n!x.P \equiv n!x.Q$
(Stru Trans) $P \equiv Q\ Q \equiv R \Rightarrow P \equiv R$	(Stru Inp) $P \equiv Q \Rightarrow n(x).P \equiv n(x).Q$
(Stru Par Assoc) $P \mid (Q \mid R) \equiv (P \mid Q) \mid R$	(Stru Rec) $P \equiv Q \Rightarrow \mathrm{rec}\ X = P \equiv \mathrm{rec}\ X = Q$
(Stru Par Comm) $P \mid Q \equiv Q \mid P$	(Stru Res Zero) $(\nu n)0 \equiv 0$
(Stru Par Zero) $P \mid 0 \equiv P$	(Stru Res Res) $(\nu n)(\nu m)P \equiv (\nu m)(\nu n)P$
(Stru Res) $P \equiv Q \Rightarrow (\nu n)P \equiv (\nu n)Q$	(Stru Res Par) $n \notin fn(Q) \Rightarrow (\nu n)(P \mid Q) \equiv ((\nu n)P) \mid Q$
(Stru Par) $P \equiv Q \Rightarrow P \mid R \equiv Q \mid R$	(Stru Res Amb) $n \neq m \Rightarrow (\nu n)m[P] \equiv m[(\nu n)P]$
(Stru Amb) $P \equiv Q \Rightarrow m[P] \equiv m[Q]$	

表 2　界程演算进程的归约关系 $\rightarrow$

Table 2　Reduction rules of mobile ambients

(Red In) $m[\mathrm{in}\ n.P \mid P] \mid n[\overline{\mathrm{in}}\ m.Q \mid Q] \rightarrow n[Q \mid Q' \mid m[P \mid P]]$
(Red Out) $n[Q \mid Q' \mid m[\mathrm{out}\ a.P \mid P]] \mid a[\overline{\mathrm{out}}\ n.R \mid R] \rightarrow n[Q \mid Q] \mid m[P \mid P] \mid a[R \mid R]$
(Red Open) $m[\mathrm{open}\ n.P \mid P] \mid n[\overline{\mathrm{open}}\ m.Q \mid Q] \rightarrow m[P \mid P' \mid Q \mid Q]$
(Red Comm) $m[n(x).P \mid P] \mid n[m!y.Q \mid Q] \rightarrow m[P\{x \leftarrow y\} \mid P] \mid n[Q \mid Q]$
(Red Amb) $P \rightarrow Q \Rightarrow n[P] \rightarrow n[Q]$
(Red Par) $P \rightarrow Q \Rightarrow P \mid R \rightarrow Q \mid R$
(Red Res) $P \rightarrow Q \Rightarrow (\nu n)P \rightarrow (\nu n)Q$
(Red Stru) $P \equiv P', P \rightarrow Q, Q \equiv Q' \Rightarrow P' \rightarrow Q'$

2 应用界程逻辑语法和语义

2.1 逻辑公式的语法

令 N 为名字的可数集合,其元素可用 $n,m,\cdots$ 表示. 令 V 为名字变量的集合,其元素一般由 $x,y,\cdots$ 等表示,用 $\eta,u,\cdots$ 等表示名字或名字变量. 令 X 为命题变量的集合,其元素由 X,Y,Z 表示. 令 $\mathrm{Action}=\{\mathrm{in}\ u,\ \overline{\mathrm{in}}\ u,\ \mathrm{open}\ u,\ \overline{\mathrm{open}}\ u,\ \mathrm{out}\ u,\ \overline{out}\ u,\ \eta?u,\ \eta!u\mid \eta,u\in V\cup N\}$ 为观察动作的集合,其元素一般用 α,β 表示.

定义1 令 Forms 表示应用界程逻辑公式构成的集合,其元素一般由 A,B 表示. 它们由下列 BNF 语法定义:

$$A,B::=T\mid\neg A\mid A\wedge B\mid\forall xA\mid 0\mid\eta\ [A]\mid A\mid B\mid$$
$$\langle\alpha\rangle A\mid\langle\cdot\rangle A\mid\eta\circledR A\mid \mathit{И}x.A\mid X\mid\nu X.A$$

在上述逻辑公式的逻辑组合符中包括命题、空间、时态、一阶量词、新鲜名字量词和名字揭示操作符. 命题公式包括了经典谓词逻辑连接词: $T,\neg,\wedge$ 和全称量词 $\forall$. 空间公式中,0 表示进程是钝性的. 公式 $\eta\ [A]$ 表示进程是一个名字为 η 的界程,其内部进程满足性质 A. $A\mid B$ 表示进程分解为二个并发进程且其中之一必须满足 A,另一方满足 B; 与 AL 逻辑不同的是,作为并发模态词"$\mid$"的并发模态副词 $\triangleright$ 被去除,而增加了用于直接观察进程潜在交互行为的动作模态公式 $\langle\alpha\rangle A$ 来表示进程如果执行动作 α 后必须满足公式 A. 公式 $\langle\cdot\rangle A$ 表示进程如果执行了一个归约动作后必须满足公式 A. 最大不动点公式 $\nu X.A$ 表示进程无限执行时满足性质 A,由此可导出最小不动点公式 $\mu X.A=\neg\nu X.\neg A$.

此外,应用界程逻辑公式还包括描述名字性质的模态词: ®和 $\mathit{И}$. $\eta\circledR A$ 表示进程可以揭示其内部名字 η. $\mathit{И}x.A$ 表示了存在相对于进程和公式 A 都是新鲜的名字 u,使得进程满足 $A\{x\leftarrow u\}$.

在公式 $\forall xA$ 和 $\mathit{И}x.A$ 中名字变量 x 在其辖域 A 中是被约束的,在公式 $\nu X.A$ 中命题变量 X 在其辖域 A 中是被约束的. 这样就确定了公式 A 中名字变量或命题变量存在的不同形式: 自由名字变量和约束名字变量,自由命题变量和约束命题变量. 令公式 A 中的自由名字集、自由名字变量集和自由命题变量分别为 $fn(A)$、$fv(A)$ 和 $fpv(A)$,并称当 $fv(A)=\varnothing$ 时公式 A 是名字闭公式. 当 $fpv(A)=\varnothing$ 时称公式是命题闭公式. 若公式 A 既是名字闭公式又是命题闭公式,则称 A 为闭公式.

在形如 $\nu X.A$ 的公式中,称在 A 中自由出现的某个命题变量 X 处于正出现位置是指: 该命题变量在 A 中出现的位置处于偶数个逻辑非操作符"$\neg$"的辖域内. 称公式 $\nu X.A$ 是良构的,如果 A 中所有自由出现的命题变量 X 都处于正出现位置. 称公式 A 是良构的,如果 A 中每一个形如 $\nu X.$ 的子公式都是良构的. 本文以下部分仅考虑良构公式.

2.2 逻辑公式的语义

给定公式 A,其逻辑语义的实质是指满足公式 A 所描述性质的所有进程组成的集合,即由公式 A 所刻画的进程构成的集合,记为$[\![A]\!]$. 因此进程集合$[\![A]\!]$需要满足如下条件: 首先,$[\![A]\!]$需要满足在结构同余下的封闭性质,即如果 $P\in[\![A]\!]$ 且 $P\equiv Q$ 则 $Q\in[\![A]\!]$. 其次,当 $P\in[\![A]\!]$ 且存在一个名字 $n\in fn(P)$ 但 $n\notin fn(A)$ 时,用名字 $m\notin\{fn(P)\cup fn(A)\}$ 来对进程 P 中的名字 n 进行替换后,对于公式 A 来说是等价的,即 $P\{n\leftarrow m\}\in[\![A]\!]$. 根据这一点,验证全称量词公式 $\forall xA$ 的满足性就不需要在无限名字集上对于每一个名字都进行检查. 进一步地,进程集$[\![A]\!]$的定义需要一个相应的有限名字集 N 作为一个支撑,满足任何 n,$m\notin N$,如果 $P\in[\![A]\!]$ 则 $P\{n\leftrightarrow m\}\in[\![A]\!]$. 为此本文引入性质集[11]的概念来刻画公式的逻辑语义.

定义2 性质集是一个满足如下条件的进程构成的集合 Φ:

(1) 如果 $P\in\Phi$ 且 $P\equiv Q$ 则 $Q\in\Phi$;

(2) 存在一个有限名字集 N,对于任何 $n,m\notin N$,如果 $P\in\Phi$ 则 $P\{n\leftrightarrow m\}\in\Phi$.

本文以下内容中,用 Φ,Ψ 表示性质集. 给定有限名字集 N,用 P_N 表示所有与集合 N 相关的所有性质集. 用 P 表示所有性质集构成的集合.

给定任意一个有限名字集 N,尊重 N 的名字置换 τ 定义为: 对于 $n\in N$ 都有 $\tau(n)=n$,一般地,用 R_N 表示尊重 N 的所有置换构成的集合. 相应地,尊重性质集 Φ 的名字置换 τ 定义为: $\tau(\Phi)=\Phi$,由此可以引出性质集的支撑概念.

定义 3　令 N 是一个名字集合,Φ 是性质集. 如果对于任一个尊重 N 的置换都相应地尊重性质集 Φ,称性质集 Φ 被 N 支撑. 进一步地,如果 N 是有限集,则称 Φ 被 N 有限支撑.

给定名字置换 τ,对性质集 Φ 的名字置换定义为: $\tau(\Phi) \triangleq \{\tau(P) \mid P \in \Phi\}$. 界程演算的进程集上进行名字置换对于结构同余是封闭的[8,10],则有 $\Phi \in P_N$ 当且仅当 $\tau(\Phi) \in P_{\tau(N)}$,由此可以得到定理 1.

定理 1　对于每一个性质集 Φ,

(1) 存在最小支撑集 $\mathrm{supp}(\Phi)$ 满足 $\Phi \in P_{\mathrm{supp}(\Phi)}$;

(2) 如果 $\mathrm{supp}(\Phi) = N$,则对于任一尊重 N 的置换 τ 有 $\mathrm{supp}(\tau(\Phi)) = \tau(N)$.

证明　略.

由定理 1 可知,对于任一置换 $\tau \in R_N$,如果需要"尊重"的集合 N 越大,则意味对置换 τ 的限制越多. 所以如果一个性质集 Φ 是被 N 支撑的且 $N \subseteq M$,则 Φ 也必然被 M 支撑. 同时,对于任何有限名字集 N,总是可以找到一个能够包含性质集 Φ 的,且被 N 支撑的最小性质集 Ψ.

由于不动点公式 $\nu X.A$ 逻辑公式的存在,子公式 A 中可能存在自由命题变量,因而需要给出用于对自由命题变量的语义进行解释的赋值函数 V.

定义 4　赋值 V 是将命题变量集 X 中的某个有限子集中的每一个命题变量 X 指派为一个性质集 Φ 的映射. 令 $D(V)$ 为赋值 V 的域. 给定公式 A,对公式 A 的赋值是指任何一个满足 $fpv(A) \subseteq D(V)$ 的赋值 V.

如果 V 是一个赋值,用 $V[X \leftarrow \Phi]$ 表示对 V 的更新,即域为 $D(V) \cup \{X\}$,将 X 指派为 Φ,而对其他 $Z \neq X$ 的命题变量的赋值仍然由 V 确定. 给定任何赋值 V,则由赋值产生的自由名字集,记 $fn(V)$,定义如下:

$$fn(V) \cup \{\mathrm{supp}(V(X)) \mid X \in D(V)\}.$$

给定公式 A 和对公式 A 的赋值 V,公式 A 在赋值 V 下的自由名字集 $fn^v(A)$ 定义为:

$$fn^v(A) \triangleq fn(A) \cup \cup \{\mathrm{supp}(V(X)) \mid X \in fpv(A)\}.$$

明显地,除了对于任何 $X \in fpv(A)$ 有 $fn^v(X) = \mathrm{supp}(V(X))$ 之外,$fn^v(A)$ 几乎就等于 $fn(A)$. 所以对于任何一个闭公式 A 来说: $fn^v(A) = fn(A)$. 在赋值 V 下的自由名字集 $fn^v(A)$ 对于解释新鲜名量词公式 $Иx.A$ 的语义是非常重要的: 当公式 A 中包含自由出现的命题变量时,$fn^v(A)$ 中的自由名字集就成为测试是否为新鲜名字的参照物.

定义 5　给定公式 A,其逻辑语义(记为 $[\![A]\!]_v$)是将某个进程集指派给名字闭公式 A 的映射,其中 V 是公式 A 的赋值,$[\![A]\!]_v$ 归纳定义如表 3 所示.

表 3　应用界程逻辑公式的指称语义

Table 3　Denotational semantics of formulas of Applied-Ambients Logic

$[\![T]\!]_v = P$	$[\![n[A]]\!]_v = \{P \mid$ 存在 $P \equiv n[Q]$ 且 $Q \in [\![A]\!]_v\}$
$[\![0]\!]_v = \{P \mid P \equiv 0\}$	$[\![A \mid B]\!]_v = \{P \mid$ 存在 $P \equiv Q \mid R$ 且 $Q \in [\![A]\!]_v$ 和 $R \in [\![B]\!]_v\}$
$[\![\neg A]\!]_v = P \setminus [\![A]\!]_v$	$[\![n \circledR A]\!]_v = \{P \mid$ 存在 $P \equiv (\nu n)Q$ 且 $Q \in [\![A]\!]_v\}$
$[\![A \wedge B]\!]_v = [\![A]\!]_v \cap [\![B]\!]_v$	$[\![Иx.A]\!]_v = \cup\, n \notin fn^v(A)\,([\![A\{x \leftarrow n\}]\!]_v \setminus \{P \mid n \in fn(P)\})$
$[\![\forall xA]\!]_v = \cap_{n \in N}[\![A\{x \leftarrow n\}]\!]_v$	$[\![\langle n!m\rangle A]\!]_v = \{P \mid$ 存在 $P \equiv n!m.Q$ 且 $Q \in [\![A]\!]_v\}$
$[\![\langle\cdot\rangle A]\!]_v = \{P \mid$ 存在 $P \to Q$ 且 $Q \in [\![A]\!]_v\}$	$[\![\langle n?m\rangle A]\!]_v = \{P \mid$ 存在 $P \equiv n(x).Q$ 且 $Q\{x \leftarrow m\} \in [\![A]\!]_v\}$
$[\![X]\!]_v = V(X)$	$[\![\langle \mathrm{cap}\ n\rangle A]\!]_v = \{P \mid$ 存在 $P \equiv \mathrm{cap}\ n.Q$ 且 $Q \in [\![A]\!]_v\}$
$[\![\nu X.A]\!]_v = \cup\{\Phi \in P \mid \Phi \subseteq [\![A]\!]_{v\{X \leftarrow \Phi\}}\}$	

从指称语义的定义中可以看出,对于没有自由出现命题变量的公式来说,对命题变量的赋值指派并不会影响到公式的指称语义,即给定公式 A,进程集 Φ 和对 A 的赋值 V,如果 $X \notin fpv(A)$,则 $[\![A]\!]_v = [\![A]\!]_{v\{X \leftarrow \Phi\}}$. 如果公式 A 是闭公式,则其指称语义并不依赖于赋值 V,在这种情况下 $[\![A]\!]_v$ 可写为 $[\![A]\!]$. 对于任意标准化进程 $P \in P$,如果 $P \in [\![A]\!]_v$,记为 $P \mid =_v A$: 表示在赋值 V 下进程 P 满足公式 A,或如果 $P \in [\![A]\!]$ 记为 $P \mid = [\![A]\!]$.

现令 $\tau(V)$ 表示名字置换 τ 作用于赋值 V,即给定一个赋值 V,且 $\tau(V)$ 的域与 V 的域是相同的,则对于所有 $X \in D(V)$,$\tau(V)(X)\ \tau(V(X))$,有定理 2:

定理 2　给定公式 A,赋值 X 和置换 τ

(1) 如果 $\tau = \{n \leftrightarrow m\}$,且 $n, m \notin fn(X)$,则 $[\![A]\!]_v = [\![A]\!]_{\tau(v)}$;

(2) $\tau(fn^v(A)) = fn^{\tau(v)}(\tau(A))$;

(3) 令 $fn^v(\nu X.A) = M$,对于所有 $\Phi \in P_M$ 有 $fn^v(\nu X.A) \subseteq fn^{v[X\leftarrow\Phi]}(A) = M$.

证明 略.

定义2 通过引入性质集的概念来描述逻辑公式的指称语义的实质,而定义5中的$[\![A]\!]_v$定义都依赖于结构同余关系$\equiv$,因而$[\![A]\!]_v$具有在结构同余下的封闭性质. 现将进一步确立$[\![A]\!]_v$和性质集Φ之间的一致性来表明$[\![A]\!]_v$的定义确实反映了逻辑公式A的指称语义的实质. 以下仅给出的相关结论,限于篇幅在此不作证明.

定理3 给定公式A,赋值V和性质集Φ,Ψ,并令$M = fn^v(A)$,有如下结论:

(1) $[\![A]\!]_v \in P_M$;

(2) 对于所有的置换τ,$\tau([\![A]\!]_v) = [\![\tau(A)]\!]_{\tau(v)}$;

(3) 如果X在A中处于正出现,且$\Phi \subseteq \Psi$,则$[\![A]\!]_{v[X\leftarrow\Phi]} \subseteq [\![A]\!]_{v[X\leftarrow\Psi]}$;

(4) 如果X在A中处于负出现,且$\Phi \subseteq \Psi$,则$[\![A]\!]_{v[X\leftarrow\Psi]} \subseteq [\![A]\!]_{v[X\leftarrow\Phi]}$.

定理4 令公式A中的命题变量X是处于正出现的,V是公式$\nu X.A$的赋值,以及$fn^v(\nu X.A) = M$,则函数$\lambda\Phi.[\![A]\!]_{v[X\leftarrow\Phi]} : P_M \to P_M$在完全格代数$P_M$上是单调增加的,则该函数有最大不动点(记为$\mathrm{gfix}P_M(\lambda\Phi.[\![A]\!]_{v[X\leftarrow\Phi]})$).

定理5 令有限名字集$M = fn^v(\nu X.A)$,则有:

$$\mathrm{gfix}P_M(\lambda\Phi.[\![A]\!]_{v[X\leftarrow\Phi]}) \subseteq [\![\nu X.A]\!]_v.$$

由上述定理5可知,一方面,如果M等于$fn^v(\nu X.A)$,则不动点必定在由集合P_M组成的格代数结构中得到. 同时在计算不动点时,选择一个较小的集合$N \subseteq fn^v(\nu X.A)$作为不动点计算的起始点进行计算并不会影响最终结果,由此有定理6.

定理6 给定公式$\nu X.A$和赋值V,如果存在一个$\subseteq$运算上的最大性质集Φ满足$\Phi = [\![A]\!]_{v[X\leftarrow\Phi]}$,则$\Phi = [\![\nu X.A]\!]_v$.

上述一系列结论中,定理3通过(1)、(2)、(3)和(4)子结论的相互归纳,证明了$[\![A]\!]_v$具有单调性;定理4确立了不动点的存在性;定理5和定理6最终给出了$[\![\nu X.A]\!]_v$就是最大不动点.

根据上述公式指称语义,"某个下一时刻"模态词$\langle\cdot\rangle$的对偶"任意下一时刻"模态词$[\,]$公式可定义为:$[\,]A = \neg\langle\cdot\rangle\neg A$,"某个时刻"模态词$\langle^*\rangle$公式和"任意时刻"模态词$[^*]$公式可分别定义为:$\langle^*\rangle A = \mu X.A \vee \langle\cdot\rangle A$和$[^*]A = \nu X.A \wedge [\,]A$. 令$\to^*$为$\to$的自反传递闭包,则可得满足关系$P \models \langle^*\rangle A$当且仅当存在$P \to^* Q$有$Q \models A$,$P \models [^*]A$当且仅当对于所有$P \to^* Q$都有$Q \models A$.

现考察应用界程逻辑公式满足性的可判定性问题. 界程逻辑[1]中并发模态副词$\triangleright$的引入是导致该逻辑公式的满足性检测不可判定的主要原因[5-7]. 本文提出的应用界程逻辑,通过去除并发模态副词$\triangleright$而引入了直接观察进程行为的模态词来描述进程交互行为,消除了这个影响因素. 虽然本文还在界程逻辑[1]基础上引入了不动点逻辑公式用于描述进程的递归行为,但通过运用性质集的概念证明了不动点逻辑公式语义的存在性,因此应用界程逻辑公式的满足性在有限控制界程演算[9,10]的进程下的模型检测是可判定的. 相应的模型检测算法限于篇幅本文不再给出,可参阅文献[10].

3 实例

应用界程逻辑通过引入动作模态词,使其具备了同时在时间、空间和动作行为这3个方面来刻画进程性质的能力. 本节给出一些实例来说明采用应用界程逻辑公式,不但可以描述形式化系统模型的基本需求性质(如系统是死锁性、可达性等),而且可以描述系统模型所需要满足的时间和空间上的特定行为性质(如系统行为在时间上或空间上的约束、处理过程安全性等). 以下给出一个采用有限界程演算进行资源传输系统建模,以及应用界程逻辑公式来描述该系统的系统需求的例子.

一个资源传输系统的工作过程描述如下:生产者 mcdonalds、pisahut 分别拥有资源 beef、pisa,消费者 student 同时请求资源 beef 和 pisa,消费者 student 首先向管理者 walmart 请求 usa 和 italy 风格的资源,而管理者 walmart 分别与生产者 mcdonalds、pisahut 进行通讯,并协调生产者和消费者之间的资源配送. 该资源传输系统的系统需求为:

P1: 系统应始终保持活动状态(无死锁情况发生).

P2: 消费者 student 最终能以 eat 方式消费 pisa 资源(资源传输的可达性).

P3: 系统的处理流程需要满足如下顺序: (1) student 首先向 walmart 发出请求; (2) walmart 从 mcdonlds 收到配送者的名字,并将配送者名字发送给消费者 student,将消费者 student 名字发送给配送者; (3) 消费者从配送者中得到资源 beef; (4) 配送者由 mcdonlds 召回(系统处理行为在时间和空间上的约束).

P4: 不能发生消费者 student 直接与 mcdonlds 发生通讯的情景(安全性).

假定已经存在一资源传输系统 R,并采用有限控制的公平界程演算进程建立了如下形式化模型:

$R \triangleq$ mcdonalds $[P_{11} \mid P_2]$ | pisahut $[P_{12} \mid P_2]$ | walmart $[P_{31} \mid P_{32}]$ | student $[P_{41} \mid P_{42}]$ | eat [rec X = student(food). X]

其中子进程 P_{11}、P_{12}、P_2、P_{31}、P_{32}、P_{41} 和 P_{42} 定义如下:

$P_{11} \doteq$ rec $X = (\nu\ n)$ walmart! n. n [out walmart. walmart(c). c! beef. $\overline{\text{open}}$ mcdonalds. X]

$P_{12} \doteq$ rec $X = (\nu\ n)$ walmart! n. n [out walmart. walmart(c). c! pisa. $\overline{\text{open}}$ pisahut. X]

$P_2 \doteq$ rec X = walmart(sender). open sender. X

$P_{31} \doteq$ rec X = student(style). mcdonalds(car). $\overline{\text{out}}$ mcdonalds. student! car. car! student. mcdonalds! car. X

$P_{32} \doteq$ rec X = student(style). pisahut(car). $\overline{\text{out}}$ pisahut. student! car. car! student. pisahut! car. X

$P_{41} \doteq$ rec X = walmart! usa. walmart(sender). sender(food). eat! food. X

$P_{42} \doteq$ rec X = walmart! italy. walmart(sender). sender(food). eat! food. X

采用应用界程逻辑公式 $F1 \sim F4$ 分别描述系统需求 $P1 \sim P4$,然后使用移动界程演算模型检测工具[10] 来验证 4 个满足关系(即验证 $R \models F1$、$R \models F2$、$R \models F3$ 和 $R \models F4$),从而可实现自动验证资源传输系统 R 是否满足上述系统需求的目标. 逻辑公式分别描述如下:

$F1 = [*]\langle\cdot\rangle T$

$F2 = \langle * \rangle$(student $[\langle$eat! pisa$\rangle T]$ | T)

$F3 = \langle * \rangle$(student $[\langle$walmart! usa$\rangle T \mid T]$ | T

$\wedge\ \langle * \rangle$car ®(walmart $[\langle$mcdonlds?car$\rangle T \mid T]$ | $T \wedge \langle * \rangle$(walmart $[\langle$student! car$\rangle\langle$car! student$\rangle T \mid T]$ | T

$\wedge\ \langle * \rangle$(student $[\langle$car?beef$\rangle T \mid T]$ | T

$\wedge\ \langle * \rangle$(mcdonlds $[\langle$open car$\rangle T \mid T]$ | car $[\langle\overline{\text{open}}$ mcdonlds$\rangle T]$ | T))))

$F4 = [*]$($\forall x. \neg$ student $[\langle$mcdonlds! $x\rangle T \mid T]$ | $T \wedge \neg$ student $[\forall x. \langle$mcdonlds?$x\rangle T \mid T]$ | T)

值注意的是,上述逻辑公式中 $F2$、$F3$ 和 $F4$ 都采用了行为模态描述了系统在某时、某地应该发生的交互行为,其中公式 $F3$ 还运用了匿名模态词®描述了配送者的名字,car 是 mcdonlds 自动生成的内部(私有)名字,但通过 walmart 这个中介成为了生产者向消费者运送资源的配送者,最后还要被 mcdonlds 回收.

上述例子表明,应用界程逻辑公式能够在时间、空间上很精细地描述系统模型的动作行为性质.

4　相关研究工作及结论

空间逻辑在模型检测中的可判定性问题是非常重要的,因为它关系到逻辑系统能否被实际使用. 一些文献[5-7] 从不同角度研究了界程逻辑的各种子逻辑公式集分别在不同界程演算模型的片断上的可判定性问题. 林惠民[12,13] 首次提出了带不动点算子的界程逻辑——谓词 mu 演算,其不动点逻辑公式可以刻画进程的无限计算行为,同时提出相应的模型检测算法. 文献[11] 首次针对 PI 演算下进行可判定的模型检测所需要的空间逻辑进行了研究,并提出了利用直接行为模态来观察进程行为性质的方法,该文的思路对本文的研究提供了有益的启示. 本文针对移动界程演算进程在空间性质和行为性质上进行可判定的模型检测需要,提出了一种可判定的在时间、空间上能够很精细地描述系统的动作行为性质的空间逻辑工具,解决了传统的界程逻辑无法实际运用于描述系统动作行为的问题,为推动移动界程演算应用于实际的移

动计算系统的形式化建模做了一定工作.

[参考文献](References)

[1] Cardelli L, Gordon A D. Anytime, anywhere: Modal logics for mobile ambients [C] // POLP 2000. Virginia Gold: ACM Press, 2000: 365-377.

[2] Cardelli L, Gordon A D. Mobile ambients [J]. Foundations of Software Science and Computation Structures(LNCS), 1998, 1378: 140-155.

[3] Hirschkoff D, Lozes É, Sangiorgi D. On the expressiveness of the ambient logic [J]. Logical Methods in Computer Science, 2006, 2(2): 1-35.

[4] Sangiorgi D. Extensionality and intensionality of the ambient logics [J]. ACM SIGPLAN Notices, 2001, 36(3): 4-13.

[5] Charatonik W. The complexity of model checking mobile ambients [C] // Proceedings of the 4th International Conference on Foundations of Software Science and Computation Structures (FoSSaCS 2001), LNCS 2030. Berlin: Springer-Verlag, 2001: 152-167.

[6] Charatonik W, Talbot J M. The decidability of model checking mobile ambients [C] // Proceedings of the 15th Annual Conference of the European Association for Computer Science Logic. Berlin: Springer, 2001: 339-354.

[7] 颜锋,陈韬略,韩婷婷,等. 空间逻辑的一个定义框架及其可判定性 [J]. 计算机科学, 2006, 33(6): 7-10.
Yan Feng, Chen Taolue, Han Tingting, et al. A definition framework of spatial logic and decedability [J]. Computer Science, 2006, 33(6): 7-10. (in Chinese)

[8] Fu Y. Fair ambients [J]. Acta Informatica, 2007, 43(8): 535-594.

[9] Charatonik W, Gordon A D, Talbot J M. Finite-control mobile ambients [C] // 11th European Symposium on Programming (ESOP 2002), LNCS2305. Berlin: Springer, 2002: 295-313.

[10] 林荣德. 移动界程演算及模型检测应用的关键问题研究 [D]. 广州: 华南理工大学计算机科学与工程学院, 2010.
Lin Rongde. Research on key issues of mobile ambients and model checking applications [D]. Guangzhou: College of Computer Science and Engineering, South China University of Technology, 2010. (in Chinese)

[11] Caires L. Behavioral and spatial observations in a logic for the π-calculus [J]. Foundations of Software Science and Computation Structures(LNCS), 2004, 2987: 72-89.

[12] Lin H. A predicate spatial logic for mobile processes [J]. Science in China Serise F: Information Sciences, 2004, 47(3): 394-408.

[13] Lin H. A predicate mu-calculus for mobile ambients [J]. Journal of Computer Science and Technology, 2005, 20(2): 95-104.

[责任编辑: 严海琳]