用于反应系统的修改时序逻辑*

贾国平 郑国梁

(南京大学计算机科学系 南京 210093)

摘要 本文提出了一个用于反应系统规范及验证的修改时序逻辑.它包含一个用于显示区分程序执行步同环境执行步的机制.环境的特性可以在系统开发时进行考虑.文中首先给出了程序的一个可复合计算模型——模块转换系统.基于此模型,给出了修改时序逻辑以及它的证明规则.本文提出的方法基于Manna-Pnueli的时序逻辑框架.经典的资源分配问题的例子用于说明此方法.最后给出了并行复合原理,它可以看成是Abadi和Lamport的关于复合假设/保证规范研究工作的具体应用.

关键词 反应系统,时序逻辑,规范,验证,模块转换系统,资源分配问题,并行复合,假设/保证规范.

中图法分类号 TP311.1

反应系统(Reactive System)是一个与其环境进行交互的系统,其目的是维持与其环境的交互而不是得到一个最终计算值.[1]反应系统中一个很重要的概念就是环境.一个反应系统是与其环境并发执行的.在反应系统的研究中,必须区分一个活动是由系统执行还是由环境执行.为了区分它们,在反应系统的行为描述中,不仅要说明状态改变,而且还必须说明每一状态改变是由系统执行所致还是由环境执行所致.反应系统中另一个非常重要的概念是复合性(Compositionality).复合性只有在与其环境交互的系统中才有意义,它也是反应系统有别于其它系统的一个主要特征.反应系统复合性的研究目前已经成为最活跃的研究领域之一,它也是并发系统自顶向下、逐步求精中最希望有的特性.

近几年来,反应系统的规范、设计和验证已经成为研究的热点,并且已经取得了很大的进展.现在已出现了许多这方面的工作,如LOTOS,UNITY,LUSTRE,TLA等.比较典型的是Manna和Pnueli的工作.[2]在此工作中,他们将时序逻辑用于反应系统的形式描述和验证.同时,也给出了一系列的用于证明系统时序性质的完全证明系统.这些应用说明时序逻辑非常适于反应系统的形式开发.

对Manna-Pnueli工作的主要反对点在于这一方法的全局性.本质上,Manna-Pnueli方法将系统视为一个自治系统.系统并不与环境进行通信.用此方法开发系统时,环境作为系统的一个组成部分,它们一起作为一个整体来研究.由此方法得到整个系统的规范后,再逐

---

* 本文研究得到博士点基金及江苏省应用基础基金资助.作者贾国平,1968年生,博士,主要研究领域为软件工程,形式化方法,时序逻辑.郑国梁,1937年生,教授,博士导师,主要研究领域为软件工程,软件开发环境.

本文通讯联系人,贾国平,上海 201206,上海贝尔电话设备制造有限公司技术开发部

本文1996-10-22收到修改稿

渐将全局规范分为环境部分和系统部分. 另外,Manna-Pnueli 方法中,原有的基本计算模型——转换系统模型并不具有可复合性. 由于这些原因,Manna-Pnueli 框架并不非常适用于反应系统(或并发系统)的层次开发.

本文基于 Manna-Pnueli 的时序逻辑框架,提出了用于反应系统的修改时序逻辑.

1 模块转换系统模型

本节首先给出程序的一个可复合计算模型——模块转换系统. 它含有一个用于区分程序执行与环境执行的机制. 我们给出了与此模型有关的一些基本概念,特别是程序行为及其计算的定义. 为了表达模块转换系统的语法,本节使用了一个基本的一阶语言,此语言中的公式称为断言.

一个模块转换系统 M 是一个四元组:$\langle V_M, \Sigma, \Theta, T\rangle$,其中 $V_M=\{u_1,\ldots,u_k\}$:状态变量的有穷集合;Σ:状态集合;Θ:初始条件;T:有穷转换集合. 每一转换 $\tau\in T$ 是一函数:$\Sigma\to 2^{\Sigma}$.

记与模块 M 相联的所有转换集合为 T_M,称为模块转换集. 它表示模块 M 所有可能的执行步. 记 T_E 为环境转换集. 它表示环境对模块 M 的所有可能的干扰. 有穷转换集合 $T=T_M\cup T_E\cup\{\tau_I\}$,其中 τ_I 为单位转换,即对每一状态 $s\in\Sigma$,$\tau_I(s)=\{s\}$.

一个转换在状态 s 下是能行的,如果 $\tau(s)\neq\varnothing$,否则 τ 在此状态下是不能行的. 如果只有单位转换 τ_I 在 s 上是能行的,一个状态 $s\in\Sigma$ 即称为终止状态. 每一转换 τ 都由一个断言来表示,记为 $\rho_\tau(V,V')$,它称为转换关系. 它将状态 $s\in\Sigma$ 与它的 τ-后继状态 $s'\in\tau(s)$ 通过无上撇号和有上撇号的状态变量联系起来. 无上撇号的状态变量引用 s 中的值. 有上撇号的状态变量引用 s' 中的值. 每一转换关系 $\rho_\tau(V,V')$ 由一能行条件 $En(\tau)$ 和一转换公式 $Trans(V,V')$ 组成,即 $\rho_\tau(V,V')=En(\tau)\wedge Trans(V,V')$,其中 $En(\tau)=\exists\ V'.\ \rho_\tau(V,V')$. 它是一状态公式,只引用 s 中的值.

我们称形如 $\sigma: s_0\xrightarrow{\tau_1}s_1\xrightarrow{\tau_2}s_2\ldots$ 的有穷或无穷行为序列是模块转换系统 M 的一个计算(Computation),如果 σ 满足:

- 初始化条件:s_0 是初始状态,即 $s_0\models\Theta$.
- 连续性:对每一状态 s_k,存在转换 $\tau\in T$,满足 $s_{k+1}\in\tau(s_k)$,其中或者 $\tau\in T_M$,表示一个模块转换步;或者 $\tau\in T_E$,表示一个环境转换步;或者 $\tau=\tau_I$,表示无状态改变.
- 勤奋性:或者 σ 包含无穷多个非 τ_I 转换,或者 σ 中包含一个终止状态.

对一个模块转换系统 M,记 $Comp(M)$ 为系统 M 的所有计算的集合.

下面给出一些定义. 我们约定 $|\sigma|$,$\sigma|_k$,$\sigma.s_k$ 及 $\sigma.\tau_k$ 分别表示计算 σ 的长度、σ 的长度为 k 的有穷序列前缀、σ 的第 k 个状态以及 σ 的第 k 个转换($k\geqslant 1$).

定义 1. 对于 2 个模块 M_1,M_2,计算 $\sigma_1\in Comp(M_1)$ 和 $\sigma_2\in Comp(M_2)$ 称为是相容的,如果它们满足:(1) $|\sigma_1|=|\sigma_2|$;(2)对所有 k,有 $\sigma_1.s_k=\sigma_2.s_k$;(3)不存在 k,使得 $\sigma_1.\tau_k\in T_{M1}$ 和 $\sigma_2.\tau_k\in T_{M2}$ 同时成立.

直观上,模块 M_1,M_2 中的两个计算是相容的,如果它们的不带标号的状态序列是相同的且它们不存在同一位置的转换,既代表模块 M_1 中的转换步又代表 M_2 中的转换步.

定义 2. 设两个模块 M_1,M_2,对它们的两个相容计算 σ_1,σ_2,定义联合计算 $\sigma_1*\sigma_2$ 为:

(Ⅰ) $|\sigma_1 * \sigma_2| = |\sigma_1|$.

(Ⅱ) $(\sigma_1 * \sigma_2).s_k = \sigma_1.s_k$.

(Ⅲ) $(\sigma_1 * \sigma_2).\tau_k = \begin{cases} \sigma_1 \cdot \tau_k, & \text{if } \sigma_1 \cdot \tau_k \in T_{M1} \\ \sigma_2 \cdot \tau_k, & \text{otherwise} \end{cases}$.

由上述两个定义，我们可以构造两个模块 M_1 和 M_2 的并行复合 $M_1 \| M_2$ 的计算模型.

定义 3. $Comp(M_1 \| M_2) = \{\sigma |$ 存在 $\sigma_1 \in Comp(M_1), \sigma_2 \in Comp(M_2), \sigma_1$ 与 σ_2 相容，满足 $\sigma = \sigma_1 * \sigma_2\}$.

定义 4. 设 μ 是一转换集合，则两个计算 σ 和 ρ 是 μ-等价的，记为 $\sigma \cong_\mu \rho$ 当且仅当对所有 $i \geqslant 0$，满足 (1) $\sigma.s_i = \rho.s_i$. (2) $\sigma.\tau_{i+1} \in \mu$ 当且仅当 $\rho.\tau_{i+1} \in \mu$.

定义 5. 设 μ 是一转换集合，一个计算集合 P 是 μ-抽象的，当且仅当对任何 μ-等价的行为 σ 和 ρ，$\sigma \in P$ 当且仅当 $\rho \in P$. 记 $Closure(P)|_\mu$ 为集合 P 在 μ-等价下的闭包.

显然，由上述定义知道，对转换集为 T_M 的模块 M，其计算集合 $Comp(M)$ 是 μ-抽象的.

由上述定义我们有如下结论.

定理 1. 设两个模块 M_1, M_2，其相联转换集合分别为 T_{M1} 和 T_{M2}，满足 $T_{M1} \cap T_{M2} = \varnothing$，则并行复合 $M = M_1 \| M_2$ 的计算 $Comp(M)$ 为：$Comp(M) = Comp(M_1 \| M_2) = Closure(Comp(M_1) \cap Comp(M_2))|_{T_M}$，其中 $T_M = T_{M1} \cup T_{M2}$.

证明：我们只证明左⊆右的蕴含关系，相反的蕴含关系由定义类似可证.

设 $\sigma \in Comp(M_1 \| M_2)$，由定义 3 可知，存在 $\sigma_1 \in Comp(M_1), \sigma_2 \in Comp(M_2)$ 且 σ_1 与 σ_2 相容，使得 $\sigma = \sigma_1 * \sigma_2$. 由相容性定义及条件 $T_{M1} \cap T_{M2} = \varnothing$，我们有 $\sigma \cong_\mu \sigma_1$ 且 $\sigma \cong_\mu \sigma_2$. 又由于 $Comp(M_1)$ 及 $Comp(M_2)$ 分别为 T_{M1}-抽象及 T_{M2}-抽象的，因此，$\sigma \in Comp(M_1)$ 且 $\sigma \in Comp(M_2)$，即 $\sigma \in Comp(M_1) \cap Comp(M_2)$. 显然有结论左⊆右. □

由定理 1 可知，并行复合 $M = M_1 \| M_2$ 的计算集合 $Comp(M)$ 为 T_M-抽象的，其中 $T_M = T_{M1} \cup T_{M2}$.

2 修改时序逻辑

本节首先简要回顾一般的时序逻辑语言，然后给出修改时序逻辑. 我们给出了修改时序逻辑语言中公式在允许行为(计算)集合上的解释.

2.1 时序逻辑

时序逻辑是一般经典逻辑的扩充. 它除了包含一般逻辑算子，如布尔联接词 ～，∧，∨，≡，→；等式算子 = 以及一阶量词∃，∀ 等外，它还包含一元时序算子：○(下一值)，□(总是)，◇(最终)及二元时序算子：W(直到)，其中算子○只可作用于项. 作用于项时，我们记 $\bigcirc t \equiv t'$，称为项 t 的下一值. 直观上，项 t' 表示经过某一转换后项 t 的值. 另外，算子 W 是主要的. 其它时序算子(包括算子□和◇)可通过它来定义，如 $\Box p \Leftrightarrow p \ W \ false$，$\Diamond p \Leftrightarrow \sim\Box\sim p$，$pWq \Leftrightarrow \Box p \vee (p \ W \ q)$ 以及 $p \triangleright q \Leftrightarrow \Box(p \rightarrow \Diamond q)$ 等. 我们将它们作为简写.

时序逻辑语言中的变量集合 V 分为两部分：一部分为动态变量或程序变量. 它不允许量词约束；另一部分为静态变量或逻辑变量. 它只用于规范目的，可以由量词约束.

一个 Kripke 结构为 $K = (S, \xi, \eta)$，其中 S 为一解释，它说明一个非空论域 $|S|$ 且指定论域 $|S|$ 中的某个元素作为语言中常元的解释，指定论域 $|S|$ 上的函数和谓词作为语言中函数

符号及谓词符号的解释;ξ 为全局变量估值函数. 它对每一变量集合中的静态变量赋予论域 $|S|$ 中的一个元素;η 为一有穷或无穷状态序列:$s_0, s_1, s_2, \ldots$,其中每一状态 s_j 是一局部变量估值函数. 它对每一变量集合中的动态变量赋予论域 $|S|$ 中的一个元素.

不包含任何时序算子(包括施用于项的下一值算子)的公式称为状态公式.

给定一(时序)公式 ω,下面归纳定义公式 ω 在结构 K 中的位置 $i \geqslant 0$(即第 i 个时刻)的可满足性(我们记为 $(\xi, \eta_i) \models \omega$,其中解释 S 是隐含表示的). 此处,我们只给出时序公式的可满足性. 状态公式情形与一阶语言中定义类似.

- $(\xi, \eta_i) \models p\ W\ q$ 当且仅当存在 j: $i \leqslant j \leqslant |\eta|$, $(\xi, \eta_j) \models q$,并且对所有 k: $i \leqslant k < j$,有 $(\xi, \eta_k) \models p$.
- $(\xi, \eta_i) \models \Box p$ 当且仅当对每一 j: $i \leqslant j \leqslant |\eta|$,有 $(\xi, \eta_j) \models p$.
- $(\xi, \eta_i) \models \Diamond p$ 当且仅当存在某一 j: $i \leqslant j \leqslant |\eta|$,使得 $(\xi, \eta_j) \models p$.

在所有状态上都成立的状态公式称为断言有效的(Assertionally Valid).

如果 $(\xi, \eta_0) \models \omega$,则称公式 ω 在结构 K 上成立,简记为 $\xi \models \omega$. 一个公式 ω 称为是可满足的(Satisfiable),如果它在某些结构 K 上成立. 一个公式 ω 称为是时序有效的(Temporally Valid),如果它在所有结构 K 上都成立.

为了应用于程序的讨论,我们常常将注意力限制在程序 P 的计算集合 $Comp(S)$ 上,即由 P 的所有计算组成的结构集合上(固定 S 及 ξ). 如果公式 ω 在 P 的所有计算上均成立,则称 ω 是 P-有效的. 每一时序有效公式对任何系统 P 在其上均是有效的.

显然,由上述定义可知,对每一(时序)公式 ω,其语义解释 $[\omega]$ 可以视为一满足此公式的序列集合,即 $[\omega] = \{\eta \mid$ 对任意全局估值 ξ,成立 $(\xi, \eta_0) \models \omega\}$

2.2 修改时序逻辑

由前面关于程序计算的定义知道,反应系统模型既包含系统转换集合又包含环境转换集合,即计算中既包含系统转换步又包含环境转换步. 因此,适用于此模型的逻辑系统也应该包含用于区分它们的公式. 一般时序逻辑语言显然不便于对程序的这类行为进行描述. 下面我们对时序逻辑语言进行修改,给出修改时序逻辑.

令 μ 表示一转换集合,$\overline{\mu}$ 表示其补集合. 假设 μ 及 $\overline{\mu}$ 均为非空. 为了得到只对 μ 中转换进行约束的公式,我们将时序算子 W 修改为 W_μ. 这种修改本质上与文献[3,4]中为了得到可复合时序逻辑而进行的修改类似. 为了便于本文应用,引入算子 $\Box_\mu$ 和 $\Diamond_\mu$. 它们可由算子 W_μ 定义,如 $\Box_\mu p \equiv p\ W_\mu false$, $\Diamond_\mu p \equiv \sim \Box_\mu \sim p$. 我们将它们作为缩写. 特别地,记算子 Δ_μ 为:$p \Delta_\mu q \equiv \Box_\mu (p \rightarrow \Diamond_\mu q)$.

修改时序逻辑中的语义结构为 $K' = (S, \xi, \sigma)$,其中 S 及 ξ 同前,分别为解释和全局变量估值函数;σ 为形如 $s_0 \xrightarrow{\tau_1} s_1 \xrightarrow{\tau_2} s_2 \ldots$ 的有穷或无穷计算序列,其中 τ_i 为转换,$\tau_i \in \mu$ 或 $\tau_i \in \overline{\mu}$. 状态 s_j 是局部变量估值函数.

修改时序逻辑中公式 ω 在语义结构 K' 中位置 $i \geqslant 0$ 的可满足性归纳定义如下(记为 $(\xi, \sigma_i) \models \omega$):

- $(\xi, \sigma_i) \models p\ W_\mu q$ 当且仅当对任何 j: $i < j \leqslant |\sigma|$, $(\xi, \sigma_{j-1}) \models p \wedge \sim q$ 且 $\sigma.\tau_j \in \mu$,有 $(\xi, \sigma_j) \models p \vee q$.

直观上,$p\ W_\mu q$ 成立当且仅当 μ 中的转换将状态 $p \wedge \sim q$ 转换为状态 $p \vee q$,即如果状态 $p \wedge \sim q$ 在 μ 中一转换之前成立,则执行此转换后,除非 q 成立,否则 p 仍然成立. 显然,此公式只对 μ 中转换进行约束.

• $(\xi,\sigma_i)\models\Box_\mu p$ 当且仅当对任何 j: $i<j\leqslant|\sigma|$, $(\xi,\sigma_{j-1})\models p$ 且 $\sigma.\tau_j\in\mu$, 有 $(\xi,\sigma_j)\models p$.

直观上, $\Box_\mu p$ 说明无 μ-转换使 p 为假.

• $(\xi,\sigma_i)\models\Diamond_\mu p$ 当且仅当存在 j: $i<j\leqslant|\sigma|$, 满足 $(\xi,\sigma_{j-1})\models\sim p$, $\sigma.\tau_j\in\mu$ 且 $(\xi,\sigma_j)\models p$.

直观上, $\Diamond_\mu p$ 说明总存在一个 μ-转换使 p 为真.

对其它(时序)公式,其可满足性与未修改情形相同.

修改时序逻辑中时序公式的有效性及 P-有效性等可由可满足性类似定义.

与一般时序逻辑类似,对每一(时序)公式 ω,在修改时序逻辑中其语义解释 $[\omega]$ 可以视为一满足此公式的计算集合,即 $[\omega]=\{\sigma|$ 对任意全局估值 ξ, 成立 $(\xi,\sigma_0)\models\omega\}$.

在修改时序逻辑中,我们只使用了时序算子 W_μ 及导出算子 $\Box_\mu$ 及 $\Diamond_\mu$. 为了增强语言的表达能力,在系统模型的状态变量集合 V_M 中加入辅助变量. 它是记录程序执行历史的状态变量. 它的值作为系统内部状态的抽象表示,并不影响系统的执行,其目的并不在于最终实现. 我们所使用的辅助变量是一种确定的辅助变量,其当前值是由其它变量的过去值及当前值来决定的. 在许多文献中它也被称为历史变量.

形式上,状态变量集合 V_M 的一个子集 V_A 是辅助变量集合当且仅当 V_A 的变量满足:(1)不出现在转换关系的能行条件中;(2)不影响非 V_A 中的状态变量值的改变.

我们不仅用辅助变量说明系统的性质,还用它来证明系统的期望性质.

需要注意的是,时序算子 W, $\Box$ 和 $\Diamond$ 将不在规范中出现,但是它们将在程序性质的证明中出现.

3　证明规则

一个 P-有效的时序公式说明了程序 P 的一个性质,即说明了由程序 P 的所有计算都满足的一个条件. 本节我们给出用于建立公式的 P-有效性的证明规则. 我们的注意力集中于一个特殊的程序模块 M,它由模块转换系统 M: $\langle V_M,\Sigma,\Theta,T\rangle$ 来说明.

由计算模型及修改时序逻辑的定义知道,推理规则 $f_1,\ldots,f_N\vdash p$ 是合理的,如果成立 $[f_1]\cap[f_2]\cap\ldots\cap[f_N]\subseteq[p]$. 所有 Manna-Pnueli 框架中的证明规则,如文献[5,6],在我们的方法中均是有效的. 特别地,对我们给出的修改时序算子 W_μ, $\Box_\mu$ 和 $\Diamond_\mu$,有下述的推理规则:

• Unless

$$\frac{\text{对所有 } \tau\in\mu,\text{有 } \{p\wedge\sim q\}\tau\{p\vee q\}}{p\ W_\mu q}$$

• Mono

$$\frac{p\ W_\mu q \qquad p\to p',\ q\to q'}{p'W_\mu q'}$$

• Union

$$\frac{p\ W_{\mu_1}q \qquad p\ W_{\mu_2}q}{p\ W_{\mu_1\cup\mu_2}q}$$

• Decomp

$$\frac{p\ W_\mu q \qquad \upsilon\subseteq\mu}{p\ W_\upsilon q}$$

• Inv

$$\frac{\Theta\to p \qquad \Box_\mu p,\Box_\mu p}{\Box p}$$

其中 p 和 q 均为状态公式, $\{p\}\tau\{q\}$ 为转换 τ 相对于公式 p 和 q 的验证条件. 它是一蕴含式: $(\rho_\tau\wedge p)\to q'$, 其中 ρ_τ 为对应于转换 τ 的转换关系, q' 是通过将 q 中每一变量用其带上撇号变量代替而得到的断言. μ(或 μ_i)为对应于模块 M(或 M_i)中的转换集合.

由上述给出的证明规则,特别地,规则 Unless 和 Mono,我们可以证明它们对证明时序公式 $p\ W_\mu q$ 的 P-有效性是(相对)完备的. 证明过程与文献[5,6]中类似. 具体可见文献[7].

由此我们得到结论,上述给出的证明规则连同标准的时序逻辑证明规则是合理的,且是(相对)完备的.

4 假设/保证规范

说明一个反应系统的一般方法是就其假设和保证部分来进行说明，即给出其假设/保证规范(A/G 规范).[8]直观上，A/G 规范形式要求如果环境满足假设 A，则系统总保证满足性质 G. 现有的许多用于反应系统开发的可复合方法都基于此 A/G 规范形式. 本节给出 A/G 规范的具体形式及其含义. 然后说明由上节得到的修改时序逻辑是如何对一个规范的系统转换部分及环境转换部分进行限制的.

设 μ 为一个转换集，$\bar{\mu}$ 为集合 μ 的补集. μ 及 $\bar{\mu}$ 均非空. A,G 为两个公式集.

定义 6. 性质 A 称为不限制 μ，如果它满足条件：$\forall\ \sigma,\forall\ k<|\sigma|,\sigma|_k\in[A]\wedge\sigma.\tau_k\in\mu\rightarrow\sigma|_{k+1}\in[A]$. 记为 A Not_Constrain μ.

直观上，性质 A 不限制 μ 当且仅当 A 中每一行为均可由仅仅执行 μ 中转换步而得到. 它对应文献[9]中的 μ-可接受(μ-Receptive)概念.

定义 7. 对于一转换集为 μ 的系统，一个 A/G 规范为一个三元组(μ,A,G)，其中 A 不限制 μ，即 A Not_Constrain μ；G 不限制$\underline{\mu}$，即 G Not_Constrain $\underline{\mu}$.

在修改时序逻辑中，我们分别通过使用下标 μ 及$\underline{\mu}$ 来对系统规范的假设及保证部分进行限制，其中$\underline{\mu}$ 及 μ 中的转换分别隐含表示环境执行步及系统执行步. 这些语法限制具体如下：对于转换集为 μ 的系统，其规范的假设部分 A 仅仅包含初始条件 Θ，$W_{\underline{\mu}}$-公式及$\triangleright_{\bar{\mu}}$-公式；而规范的保证部分仅仅包含 W_{μ}-公式及$\triangleright\mu$-公式. 显然，由此限制而得到的 A/G 规范(μ,A,G)满足：A Not_Constrain μ 及 G Not_Constrain $\bar{\mu}$.

定义 8. 一个转换集为 μ 的程序 P 满足 A/G 规范(μ,A,G)，即 P sat (μ,A,G)，如果对 $\forall\ \sigma\in Comp(P),\sigma\in[A]\rightarrow\sigma\in[G]$且$\forall\ k<|\sigma|,\sigma|_k\in[A]\rightarrow\sigma|_k\in[G]$.

5 例子：资源分配问题

本节给出一个系统的资源分配问题作为本文方法的说明，其中系统由一个分配程序 Allocator 和 N 个相互竞争使用资源的用户程序 $C_i(i=1,\ldots,N)$组成(如图 1).

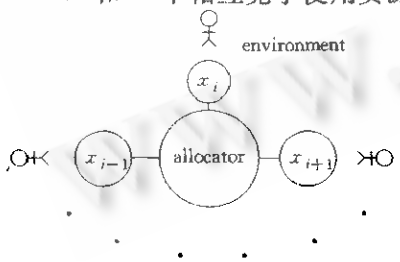

图1 资源分配问题

在我们的资源分配问题中，分配程序和用户程序之间通过共享状态变量 $x_i(i=1,\ldots,N)$进行通信. 为了使问题的描述更加形象，我们采用哲学家问题中的思考(T)、饥饿(H)和用餐(E)三个概念. 我们让每一变量 x_i 在集合$\{T,H,E\}$上变化. 对 $i=1,\ldots,N$，转换 $T\rightarrow H\equiv(x_i=T\wedge x_i'=H)$表示用户程序 C_i 请求资源. 转换 $H\rightarrow E\equiv(x_i=H\wedge x_i'=E)$表示用户程序 C_i 允许使用资源. 转换 $E\rightarrow T\equiv(x_i=E\wedge x_i'=T)$表示用户程序 C_i 释放占有的资源. 设所有共享通信变量 x_i 的初始值为 T. 我们需要解决的问题是每一请求资源的用户程序最终被允许使用资源.

上述资源分配问题是一个经典的用于说明分布式程序开发的例子. 它已经用于许多方法的说明和描述. 作为本文方法的说明及与其它方法的比较，我们对此例子进行描述. 此处，

我们集中于分配程序 Allocator 的开发. 将 N 个用户程序 C_i 与分配程序进行区分. 显然,分配程序 Allocator 是一个典型的反应系统. 它的环境由 N 个用户程序 C_i 组成. 它与用户程序 C_i 具有同样的作用.

设 $Grant\_number=\sum_{i=1}^{N}(x_i=E)$,其中如果用户程序 C_i 占有资源,则 $(x_i=E)=1$,否则 $(x_i=E)=0$,即 $Grant\_number$ 表示占有资源的用户程序 C_i 的个数.

对于分配程序 Allocator 规范中的环境假设部分,首先说明每一用户程序 C_i 初始都处于非请求资源状态,即 $x_i=T$. 环境假设部分中,转换 $T\to H$ 及 $E\to T$ 是允许的,然而转换 $H\to E$ 是不允许的,即环境可以申请资源,也可以释放资源,但是环境部分不可以自己批准占有资源. 另外,环境假设应该确保每一用户程序 C_i 最终会释放资源.

对分配程序 Allocator 规范中的保证部分,首先保证资源的互斥使用. 在分配程序中,唯一允许的转换为 $H\to E$,即分配程序的唯一目的是给请求资源的用户程序分配资源. 另外,保证部分应该确保每一请求资源的用户程序 C_i 最终可以使用资源.

由上讨论可知,分配程序 Allocator 的 A/G 规范如下:

环境假设$(A)(i=1,\dots,N)$	系统保证(G) $(i=1,\dots,N)$
$x_i=T$	$\Box_\mu(Grant\_number\leqslant 1)$
$\Box_{\overline{\mu}}(x_i=H)$	$\Box_\mu(x_i=T)$
$(x_i=T)W_{\overline{\mu}}(x_i=H)$	$\Box_\mu(x_i=E)$
$(x_i=E)W_{\overline{\mu}}(x_i=T)$	$(x_i=H)W_\mu(x_i=E)$
$(Grant\_number=1)\triangleright_{\overline{\mu}}(Grant\_number=0)$	$(x_i=H)\triangleright_\mu(x_i=E)$

6　复合原理

反应系统的可复合开发方法要求有一个复合原理,它能够通过分别验证程序 P_1 和 P_2 的规范来验证并行程序 $P_1\parallel P_2$ 的规范. Abadi 和 Lamport 给出了一种用于 A/G 规范的语义复合规则[9]:

$$\frac{P_1\ \text{sat}\ (A_1,G_1),\quad [A]\cap[G_1']\cap[G_2']\subseteq[A_1]\cap[A_2]}{P_2\ \text{sat}\ (A_2,G_2),\quad [A]\cap[G_1]\cap[G_2]\subseteq[G]}$$

$$P_1\parallel P_2\ \text{sat}\ (A,G)$$

其中$[S]$表示规范 S 允许的行为集. G_1^- 和 G_2^- 分别是由规范 (A_1,G_1) 及 (A_2,G_2) 的保证部分 G_1 和 G_2 导出的安全性. 前提$[A]\cap[G_1^-]\cap[G_2^-]\subseteq[A_2]$的含义如下:由于程序 P_2 的环境是由程序 P_1 及整个环境(即并行程序 $P_1\parallel P_2$ 的环境)组成,因此对程序 P_2 的环境假设 A_2 一定能够由程序 P_1 的保证部分以及整个环境的假设 A 导出. 保证部分 G_2^- 用于将蕴含限制到程序 P_2 的可达状态集合上. 前提$[A]\cap[G_1^-]\cap[G_2^-]\subseteq[A_1]$含义类似. 前提$[A]\cap[G_1]\cap[G_2]\subseteq[G]$含义为:在整个环境的假设 A 下,并行程序 $P_1\parallel P_2$ 的保证部分一定能够由程序 P_1 和 P_2 的保证部分蕴含.

复合原理成立的前提是:①环境假设 A,A_1 和 A_2 是安全性;②初始条件只出现于假设部分;③假设部分只对环境转换进行约束;④保证中的安全性部分只对系统转换进行约束.

显然,由前两节讨论知道,由修改时序逻辑得到的规范是能够在 Abadi-Lamport 的语

义模型中进行解释的.它们也是足以用于表达规范的假设及保证部分的.为了应用上述语义复合规则,我们此处只需对 A/G 规范形式进行转化.

引入二元算子⇒,其语义含义为:$[p⇒q]=\{\sigma|\sigma\in[p]\rightarrow\sigma\in[q]\}$.对 A/G 规范进行转化得(μ,A',G'),其中 A' 仅仅包含初始条件 Θ 及 $W_{\bar{\mu}}$-公式;而 G' 包含 W_{μ}-公式及⇒公式,其中⇒公式的前项为$\triangleright_{\bar{\mu}}$-公式,后项为$\triangleright_{\mu}$-公式.由文献[9]中的定理1可知转化而得的规范与原规范形式是实现等价的.显然,转化后的 A/G 规范(μ,A',G')满足:A' Not_Constrain μ 及 G' Not_Constrain $\bar{\mu}$.又由安全性定义知,A' 是安全性.因此,对 A/G 规范形式的上述语法约束是满足语义并行复合原理中的前提假设的.

下面给出复合原理,其中上述语义复合规则中的语义前提(即集合包含)可以由我们的推理证明($\vdash$)所代替.我们的并行复合原理成立的前提是第1节的定理1.

定理2. 设 μ,μ_1 及 μ_2 为转换集合,满足 $\mu=\mu_1\cup\mu_2,\mu_1\cap\mu_2=\varnothing$.设 A,A_1 及 A_2 是安全性且初始条件只出现在它们之中.G_1^s,G_2^s 是分别由性质 G_1 和 G_2 得到的安全性,满足 $G_1\vdash G_1^s,G_2\vdash G_2^s$,则

$$\frac{\begin{array}{ll}P_1\ \mathrm{sat}\ (\mu_1,A_1,G_1), & A,G_1^s,G_2^s\vdash A1, \quad A,G_1',G_2'\vdash A_2\\ P_2\ \mathrm{sat}\ (\mu_2,A_2,G_2), & A,G_1,G_2\vdash G\end{array}}{P_1\|P_2\ \mathrm{sat}(\mu,A,G)}$$

7 结束语

7.1 与相关工作的比较

本文给出的方法其最大的优点是时序逻辑规范在程序的一个可复合计算模型上进行解释.本文的工作可以看成是 Abadi 和 Lamport 的关于 A/G 规范可复合性研究在 Manna-Pnueli 时序框架中的具体应用.本文方法保持了 Manna-Pnueli 框架中原有的推理规则及证明方法.现有的许多时序证明系统都可重用于此规范推理.

文献[10]提出了一种用于反应系统的可复合时序逻辑方法.此方法通过在时序逻辑中加入位置命题来区分规范中的环境部分和系统部分.为了增强时序逻辑的表达能力,文献[10]中加入了新的一类时序算子 *chop* 以及 *chop* *.本文所给出的修改时序逻辑,其本质特征是显示地使用下标来区分一个规范的环境部分和系统部分.其基本思想都是为了使时序逻辑适于模块的说明.为了增强语言的表达能力,我们使用了辅助变量.辅助变量的引入使系统的描述更加直观、自然.文献[10]中,由于引入的时序算子 *chop* 以及 *chop* * 的语义非常复杂,阻止了对系统期望性质的直观理解,因此很难在实际中得到应用.

最近 Abadi 和 Lamport 提出了一种用于反应系统(或开放系统)的 A/G 规范的复合方法.[11]他们的方法基于 TLA.所给出的证明规则及复合原理均是在此逻辑中讨论的.本文方法同此方法本质不同在于本文的规范并不具有 TLA 的基本形式.本文说明了在系统转换上的约束的合取式,而不是允许系统转换的析取式.

Jones 最近提出了一种用于复合规范的面向对象方法.[12]在此方法中,由于对象并无共享变量,因此程序执行中由环境对共享变量进行修改而引起的对系统的干扰就受到限制,从而减少了由此干扰而引起的复杂性.本文的方法需要对许多这类干扰进行推理证明,而对它

们的推理是非常困难的.

Collette 最近提出一种用于 A/G 规范的复合原理.[4]他的方法基于 Unity 框架. 另外值得提出比较的是此方法利用了 Unity 逻辑的简明性,保持了 Unity 证明系统. 本文提出的方法基于 Manna-Pnueli 的时序逻辑框架. 它允许我们利用现有的大多数完备时序证明系统.

7.2 进一步工作

本文提出的方法仅仅是对反应系统开发的一个初步的工作. 它本质上是一种公理化方法,保持了规范的合取特性,即需求可以简单地通过加入到合取式中而得到. 但是本文方法有一个很大的缺陷,即由它很容易写出不可实现的规范,特别当规范中包含许多转换(或活动)时. 我们进一步的工作包括找到一种求精方法. 由本文方法得到的规范作为起始点,利用求精方法进行精化直到系统的转换必须给出,然后再利用现有的用于基于活动规范的求精方法对它们继续进行形式开发. 另外,如何减少推理证明的复杂性,增强简明性,如使用面向对象方法,也是我们今后需要进一步研究的工作.

参考文献

1 Harel D, Pnueli A. On the development of reactive systems. In: Apt K P ed. Logics and Models of Concurrent Systems, NATO ASI Series F13, Berlin: Spinger-Verlag, 1985. 477～498.

2 Manna Z, Pnueli A. The temporal logic of reactive and concurrent system: specification. New York: Spinger-Verlag, 1991.

3 Abadi M, Lamport L. An old-fashioned recipe for real time. ACM Trans. on Prog. Lang. and Sys., 1994, **16(5)**: 1543～1571.

4 Collette P. Composition of assumption-commitment specifications in a Unity style. Sci. of Comput. Prog., 1994, **23**:107～125.

5 Manna Z, Pnueli A. Adequate proof principles for invariance and liveness properties of concurrent systems. Sci. Comput. Prog., 1984, **32**:257～289.

6 Manna Z, Pnueli A. Completing the temporal picture. Theor. Comp. Sci., 1991, **83(1)**:97～130.

7 贾国平. 时序逻辑研究:模型、方法及应用[博士论文]. 南京大学研究生院, 1996.

8 Jones C B. Specification and design of (parallel) programs. In: Mason R E A ed. Information Processing 83, IFIP, North-Holland: Elsevier Science Publishers, 1983. 321～332.

9 Abadi M, Lamport L. Composing specifications. ACM Trans. on Prog. Lang. and Sys., 1993, **15(1)**:73～132.

10 Barringer H, Kuiper R, Pnueli A. Now you may compose temporal logic specifications. In: Proc. of 16th ACM Symp. on Theo. of Comput., 1984. 51～63.

11 Abadi M, Lamport L. Open systems in TLA. In: Proc. of 13th Annual ACM Symp. on Princ. of Dist. Comput., Los Angeles, CA, 1994. 81～90.

12 Jones C B. Reasoning about interference in an object-based design method. In: Woodcock J C P, Larsen P G eds. FME'93: Industrial Strength Formal Methods, Lec. Notes in Comp. Sci. 670, Berlin: Springer-Verlag, 1993. 1～17.

A MODIFIED TEMPORAL LOGIC FOR REACTIVE SYSTEMS

JIA Guoping　ZHENG Guoliang

(Department of Computer Science　Nanjing University　Nanjing　210093)

Abstract This paper presents a modified version of temporal logics for the specification and verification of reactive systems. It includes a mechanism to explicitly distinguish program steps from environment steps and the characteristics of the environment can be taken into account during the development of system. A compositional computation model of programs——modular transition system is firstly given. Then based on this model, a modified temporal logic and its proof rules are presented. The proposed approach is used within Manna-Pnueli's temporal logic framework. The classical example of the Resource Allocator is used to illustrate the approach. At the end of the paper, a parallel composition principle is proposed, it can be viewed as an application of Abadi and Lamport's works on the composing assumption/guarantee specifications.

Key words Reactive system, temporal logic, specification, verification, modular transition system, resource allocator, parallel composition, assumption/guarantee specification.

Class number TP311.1