中国科学院研究生院课程讲稿

§2 程序与系统模型

程序的正确从广义上说就是包含该程序的系统的运行过程满足给定的性质。要说明程序的正确，我们需要描述系统的这种运行过程，即系统的运行模型，简称为系统模型。本章介绍系统模型。

§2.1 流程图程序

流程图程序 是一阶逻辑的扩充。 流程图程序主要有两种语句。一种是赋值语句。一种是条件语句。这些语句称为指令。 给定 $B=(F,P)$ 。我们增加以下两个集合的符号：

$$\begin{array}{ll} \text{辅助符号集} & \{:=,:,goto,if,else\} \\ \text{标号符号集} & \{beg,end,l_1,l_2,...,\} \end{array}$$

用 LB 表示标号集合。定义 (B,V) 上的流程图程序的两种指令如下：

l_1: $(x_1,...,x_n):=(t_1,...,t_n)$ goto l_2
l_1: if (e) goto l_2 else goto l_3

其中 $l_1,l_2,l_3\in LB$ 为标号， $l_1\neq end$, $t_1,...,t_n\in T_B$ 且 $\bigcup_{i=1}^n Var(t_i)\subseteq V$, $e\in QFF_{(B)}$ 且 $Var(e)\subseteq V$, $x_1,...,x_n\in V$ 且对 $1\leq i<j\leq n$, $x_i\neq x_j$ 。 冒号左边的标号称为被定义标号。冒号右边的句子称为标号定义。

Definition 2.1 一个 (B,V) 上的结流程图程序 T 为满足以下条件的指令集合。

标号 beg 必须有定义
标号 end 不被定义
出现在标号定义中的除 end 外的标号必须有定义
每个标号最多被定义一次

给定 $B=(F,P)$ 和变量集合 V 。 (B,V) 上流程图程序的集合记为 $\mathcal{L}_2^{(B,V)}$ 。

状态

流程图程序的运行状态有两个部分：即标号和变量状态。程序的运行状态集合为 $LB\times\Sigma$ 。

运行

给定 B 的一个解释 I 。给定指令 t 。 $(l_i,\sigma_i)\overset{t}{\Rightarrow}(l_{i+1},\sigma_{i+1})$ 当且仅当以下一种情况成立。

t 为 l_i: $(v_1,...,v_n):=(t_1,...,t_n)$ goto l_{i+1} ,
且 $\sigma_{i+1}=\sigma_i[v_1/I(t_1)(\sigma_i)]\cdots[v_n/I(t_n)(\sigma_i)]$
t 为 l_i: (if (e) goto l' else goto l'' ,
$\sigma_{i+1}=\sigma_i$ 且 $\sigma_i\models_I e$ 则 $l_{i+1}=l'$ ， 否则 $l_{i+1}=l''$

给定流程图程序 T 和 B 的一个解释 I 。我们定义流程图程序的运行状态转换关系 $\overset{T}{\Rightarrow}$ 如下。 $(l_i,\sigma_i)\overset{T}{\Rightarrow}(l_{i+1},\sigma_{i+1})$ 当且仅当以下一种情况成立。

存在指令 $t \in T$ 使得 $(l_i, \sigma_i) \overset{t}{\Rightarrow} (l_{i+1}, \sigma_{i+1})$ 或
不存在这样的指令且 $(l_i, \sigma_i) = (l_{i+1}, \sigma_{i+1})$ 。

以上的第二项是强调程序终止后状态保持不变。没有实质意义。 在不引起混淆的时候，我们将 $\overset{T}{\Rightarrow}$ 中的 T 省略写成 $\Rightarrow$ 。$\{(l, \sigma) \mid (l_0, \sigma_0) \overset{*}{\Rightarrow} (l, \sigma)\}$ 即是由 (l_0, σ_0) 可达的状态的集合。 程序在初始变量状态为 σ 时的运行终止，当且仅当存在 σ' 使得 $(beg, \sigma) \overset{*}{\Rightarrow} (end, \sigma')$ 。

程序性质

程序的正确性描述可以是初始状态和终止状态的关系，以及程序是否终止。 设 φ 为程序 T 的初始状态满足的条件， ψ 为对程序 T 的终止状态的要求。 φ 称为程序的前断言， ψ 称为程序的后断言。

T 对于 φ 和 ψ 的部分正确性表示为 $\varphi(\sigma) \rightarrow ((beg, \sigma) \overset{*}{\Rightarrow} (end, \sigma') \rightarrow \psi(\sigma'))$ 。记作 $\models_I \{\varphi\} T \{\psi\}$ 。

T 对于 φ 和 ψ 的完全正确性表示为 $\varphi(\sigma) \rightarrow ((beg, \sigma) \overset{*}{\Rightarrow} (end, \sigma') \wedge \psi(\sigma'))$ 。记作 $\models_I [\varphi] T [\psi]$ 。

§2.2 结构化循环语句程序

结构化循环语句程序是一阶逻辑的扩充。 重要程序要素有赋值、顺序复合、条件语句和循环语句。 给定函数符号集 F 和谓词符号集 P ，我们在 $B = (F, P)$ 上建立一个一阶逻辑，并增加以下集合的符号：

$$\{:=, ;, , if, then, else, fi, while, do, od\}$$

Definition 2.2 一个 (B, V) 上的结构化循环语句程序是一个字符串，其集合 S 定义如下：

$$S \quad ::= \quad x := t \mid S; S \mid \mathit{if}\ e\ \mathit{then}\ S\ \mathit{else}\ S\ \mathit{fi} \mid \mathit{while}\ e\ \mathit{do}\ S\ \mathit{od}$$

其中 $x \in V$ 为 V 中的变量， $t \in T_B$ 是一个不带量词、变量在 V 中的 B 上的项， $e \in QFF_B$ 是一个不带量词、变量在 V 中的 B 上的公式。

给定 $B = (F, P)$ 和变量集合 V 。 (B, V) 上的结构化循环语句程序的集合记为 $\mathcal{L}_1^{(B,V)}$ 。

状态

程序的运行状态有两个部分，即程序运行的剩余部分和变量状态。用 ϵ 表示程序剩余部分为空。 设 Σ 为变量状态的集合。程序的运行状态集合为 $(\mathcal{L}_1^{(B,V)} \cup \{\epsilon\}) \times \Sigma$ 。

运行

一个程序的语义取决于 B 的解释。 给定 B 的一个解释 I 。程序的迁移关系 $\Rightarrow$ 为 $(\mathcal{L}_1^{(B,V)} \times \Sigma) \times ((\mathcal{L}_1^{(B,V)} \cup \{\epsilon\}) \times \Sigma)$ 的一个子集。 $(S_0, \sigma_0) \Rightarrow (S_1, \sigma_1)$ 当且仅当以下一项成立。

S_0	条件	S_1	σ_1
$x := t$		ϵ	$\sigma_0[x/I(t)(\sigma_0)]$
$x := t; S$		S	$\sigma_0[x/I(t)(\sigma_0)]$
if (e) then S else S' fi	$\sigma_0 \models_I e$	S	σ_0
if (e) then S else S' fi	$\sigma_0 \not\models_I e$	S'	σ_0
if (e) then S else S' fi; S''	$\sigma_0 \models_I e$	$S; S''$	σ_0
if (e) then S else S' fi; S''	$\sigma_0 \not\models_I e$	$S'; S''$	σ_0
while e do S od	$\sigma_0 \models_I e$	S; while (e) do S od	σ_0
while e do S od	$\sigma_0 \not\models_I e$	ϵ	σ_0
while e do S od; S'	$\sigma_0 \models_I e$	S; while (e) do S od; S'	σ_0
while e do S od; S'	$\sigma_0 \not\models_I e$	S'	σ_0

$\{\sigma \mid (T, \sigma_0) \overset{*}{\Rightarrow} (T', \sigma)\}$ 即是程序 T 在初始变量状态为 σ_0 时运行可到达的变量状态。

程序 T 在初始变量状态为 σ 时的运行终止，当且仅当存在 σ' 使得 $(T, \sigma) \overset{*}{\Rightarrow} (\epsilon, \sigma')$ 。

程序性质

程序的正确性描述可以是初始状态和终止状态的关系，以及程序是否终止。设 φ 为程序 T 的初始状态满足的条件， ψ 为对程序 T 的终止状态的要求。 φ 称为程序的前断言， ψ 称为程序的后断言。

T 对于 φ 和 ψ 的部分正确性表示为 $\varphi(\sigma) \to ((T, \sigma) \overset{*}{\Rightarrow} (\epsilon, \sigma') \to \psi(\sigma'))$ 。记作 $\models_I \{\varphi\}T\{\psi\}$ 。

T 对于 φ 和 ψ 的完全正确性表示为 $\varphi(\sigma) \to ((T, \sigma) \overset{*}{\Rightarrow} (\epsilon, \sigma') \wedge \psi(\sigma'))$ 。 记作 $\models_I [\varphi]T[\psi]$ 。

§2.3 一阶迁移系统 (FTS)

FTS 是一阶逻辑的扩充。 FTS 所描述的迁移关系主要有两个要素：条件和赋值。除此之外，系统还包含初始条件的描述。

给定常元和函数符号集合 F ，谓词符号集合 G 。记 $B = (F, G)$ 上一阶逻辑公式集合为 $\mathcal{L}_B$ ， B 上的项的集合为 T_B 。记 $\mathcal{L}_B$ 中不带量词的公式集合为 QFF_B 。 在一阶逻辑的基础上，我们增加符号 $\{\longrightarrow, :=\}$ 。给定 B 和变量集合 V 。定义 (B, V) 上的迁移如下：

$$p \longrightarrow (v_1, v_2, ..., v_n) := (e_1, e_2, ..., e_n)$$

其中 $p \in QFF_B$ 为一个不带量词的公式且 $Var(p) \subseteq V$ ， $v_1, v_2, ..., v_n \in V$ 为变元， $e_1, e_2, ..., e_n \in T_B$ 为 B 上的项且 $\bigcup_{i=1}^{n} Var(e_i) \subseteq V$ 。一个迁移表示系统的一个原子动作。

Definition 2.3 *一个 (B, V) 上的 FTS 是一个二元组*

$$(T, \Theta)$$

其中 T 为 (B, V) 上迁移的有限集合， $\Theta \in QFF_B$ 为初始条件，是一个不带量词、变量在 V 中的 B 上的公式。

B 的解释确定了迁移系统描述中常元符号、函数符号和谓词符号的含义。 变量的赋值 σ 代表了系统运行中变量在某一时刻的状态。因此我们也称 σ 为状态。状态的集合为 Σ 。 以下假设 B 的解释 $I = (D, I_0)$ 为已经给定。

状态空间

系统的状态空间为 V 中变量取值的组合 $\Sigma = \{\sigma \mid \sigma(x) \in D, x \in V\}$ 。

运行

迁移 $\varphi \longrightarrow (v_1, v_2, ..., v_n) := (e_1, e_2, ..., e_n)$ 在状态 σ 可执行，当且仅当 $\sigma \models_I \varphi$ 。 设 t 是 $\varphi \longrightarrow (v_1, v_2, ..., v_n) := (e_1, e_2, ..., e_n)$ 为一个迁移。我们用 $\sigma \xrightarrow{t} \sigma'$ 表示 t 在状态 σ 可执行且 $\sigma' = \sigma[v_1/I(e_1)(\sigma_i)] \cdots [v_n/I(e_n)(\sigma_i)]$ 。 一阶迁移系统 (T, Θ) 上的一个运行是状态集 Σ 上的一个无穷序列

$$\sigma_0\sigma_1\sigma_2\sigma_3.....$$

其中 $\sigma_0 \models_I \Theta$ 且对任意 $i \geq 0$,

在状态 σ_i	存在可执行迁移 t 且 $\sigma_i \xrightarrow{t} \sigma_{i+1}$
在状态 σ_i	不存在可执行迁移且 $\sigma_{i+1} = \sigma_i$

以上的第二项是为了避免当系统运行终止或死锁时，系统的运行只能是有穷状态序列的情况。这样我们只需讨论无穷状态序列，而不必分两种情况来讨论。

可达状态

我们用 $\sigma \rightarrow \sigma'$ 表示存在 $t \in T$ 使得 $\sigma \xrightarrow{t} \sigma'$ 。状态集合 A 的可达状态集合 $rh(A)$ 为 $\{\sigma' \mid \sigma \xrightarrow{*} \sigma', \sigma \in A\}$ 。系统的可达状态集合 $rh(\Theta)$ 为 $\{\sigma' \mid \sigma \xrightarrow{*} \sigma', \sigma \models_I \Theta\}$ 。

安全性质

系统的安全性质可用逻辑公式表示。系统满足安全性质 φ ，当且仅当系统的每个运行中的每个状态都满足 φ 。即 $\forall \sigma \in rh(\Theta).\sigma \models_I \varphi$ 。

§2.4 Kripke 结构和标号 Kripke 结构

在 FTS 的描述中，系统运行的状态空间以及系统的运行由解释 I 所决定。明确的状态空间以及状态之间关系的描述有助于简化系统性质的检验。

§2.4.1 Kripke 结构 (KS)

以 FTS 为出发点，状态空间即是 Σ 。除了状态空间之外，描述系统运行的基本要素还有状态之间的迁移关系，以及系统的初始状态。 我们把状态空间，迁移关系和初始状态集组合成 Kripke 结构，定义如下。

Definition 2.4 *一个 Kripke 结构是一个三元组*

$$\langle S, \Delta, I \rangle$$

其中 S 为状态集合， $\Delta \subseteq S \times S$ 为 S 上的完全迁移关系， $I \subseteq S$ 为初始状态集。

路径和运行

我们用 $s \rightarrow s'$ 表示存在从 s 到 s' 的迁移，即 $(s, s') \in \Delta$ 。 Kripke 结构上的一条有穷路径是状态集 S 上的一个有穷序列

$$s_0s_1s_2...s_n$$

其中对任意 $0 \leq i \leq n-1$, $s_i \rightarrow s_{i+1}$ 。

Kripke 结构上的一条无穷路径是状态集 S 上的一个无穷序列

$$s_0s_1s_2...$$

其中对任意 $i \geq 0$, $s_i \rightarrow s_{i+1}$ 。

Kripke 结构上的一个运行是 Kripke 结构上的一条无穷路径

$$s_0 s_1 s_2 ...$$

其中 $s_0 \in I$ 。

可达状态

用 $\xrightarrow{*}$ 表示 $\rightarrow$ 的传递自反闭包。状态集合 A 的可达状态集合 $rh(A)$ 为 $\{\sigma' \,|\, \sigma \xrightarrow{*} \sigma', \sigma \in A\}$ 。系统的可达状态集合为 $rh(I)$ 。

安全性质

系统的安全性质可用状态集合表示。系统满足安全性质 A，当且仅当系统的每个运行中的每个状态都在 A 中。即 $rh(I) \subseteq A$ 。 安全性质的对偶性质是可达性。状态集合 A 可达，当且仅当系统的某个运行中的某个状态在 A 中。即 $rh(I) \cap A \neq \emptyset$ 。

可达性分析算法

给定一个状态集合 $S_0 \subseteq S$ 。定义 error_state(q) 为 $q \notin S_0$ 。 系统满足该安全性质 S_0 的分析可以由以下 $S - S_0$ 状态的可达性分析算法来实现。 $S - S_0$ 状态可达即系统不满足安全性质 S_0 。

```
start()
{
    W = A = {};
    for each initial state s, if (s is not in A) { add s to W; dfs(); }
}
```

```
dfs()
{
    q = last element from W;
    add q to A;
    if (error_state(q)) report("reachable");
    else for each successor state s of q if (s is not in A) { add s to W; dfs(); }
    delete q from W;
}
```

算法性质

(1) 给定一个 Kripke 结构和一个状态 q_0，即 error_state(q)=true 当且仅当 $q = q_0$，算法报告 "reachable"当且仅当该状态式可达的。

(2) 给定一个 Kripke 结构和一个状态 q_0，则 q_0 的可达性判定是线性的。

§2.4.2 标号 KS

由于 KS 对于系统的描述过于简化， KS 作为软件模型，用状态集表示性质显得很复杂。如果表示性质的集合很大，则其检查就很复杂。为了表达和计算的简单，可用命题表示性质。为了能够用命题表示性质，

我们必须将结构中的状态和什么公式在哪些状态上成立联系起来。为了定义上的简单，我们只考虑在每个状态上某些简单命题是否成立。

Definition 2.5 给定一个原子命题集合 AP 。一个 AP 上的标号 KS 是一个四元组

$$\langle S, \Delta, I, L\rangle$$

其中 $\langle S, \Delta, I\rangle$ 为 *Kripke* 结构， $L : S \to 2^{AP}$ 为状态集合到 AP 的幂集的映射。

在这样的一个结构中， $L(s)$ 表示在 s 上所有 s 成立的原子命题的集合。原子命题 p 在 s 上成立当且仅当 $p \in L(s)$ ，其它命题逻辑公式的成立与否遵从命题逻辑的解释。

记原子命题集合 AP 上所有命题公式的集合为 $\mathcal{L}_{AP}$ 。我们可以用命题逻辑的公式表示性质。给定一个 AP 上的标号 Kripke 结构 $\langle S, \Delta, I, L\rangle$ ，一个状态 $s \in S$ 满足性质 $\varphi \in \mathcal{L}_{AP}$ 记作 $M, s \models \varphi$ ，其定义如下： $M, s \models \varphi$ 成立当且仅当：

$M, s \models p$	若 $p \in AP$ 且 $p \in L(s)$ 。
$M, s \models \neg\psi$	若 $M, s \not\models \psi$ 。
$M, s \models \psi_0 \vee \psi_1$	若 $M, s \models \psi_0$ 或 $M, s \models \psi_1$ 。
$M, s \models \psi_0 \wedge \psi_1$	若 $M, s \models \psi_0$ 且 $M, s \models \psi_1$ 。
$M, s \models \psi_0 \to \psi_1$	若 $M, s \models \psi_0$ 则 $M, s \models \psi_1$ 。
$M, s \models \psi_0 \leftrightarrow \psi_1$	若 $M, s \models \psi_0$ 当且仅当 $M, s \models \psi_1$ 。

一个公式对应于一个状态集合。即 φ 对应于状态集合 $\{s \| M, s \models \varphi\}$ 。我们用 $[[\varphi]]$ 表示状态集合 $\{s \| M, s \models \varphi\}$ 。

安全性质

系统的安全性质可用公式表示。系统满足安全性质 φ ，当且仅当系统的每个运行中的每个状态都满足 φ 。即 $\forall s \in rh(I).(M, s \models \varphi)$ ，即 $rh(I) \subseteq [[\varphi]]$ 。 那么 error_state(q) 即 $M, s \not\models \varphi$ 或 $q \notin [[\varphi]]$ 。

标号 KS 与一阶迁移系统

KS 可由 FTS 生成。若 KS 由 FTS 直接生成， KS 的状态空间的大小由 FTS 的变量及其取值范围决定。给定 $B = (V, F, P)$ 上的 FTS(T, Θ) 。设 $V = \{v_1, ..., v_n\}$ 且令 $|v|$ 为 v 可能取值的个数。

状态空间的大小为 $\|v_1\| \cdot \|v_2\| \cdots \|v_n\|$ 。 若用 n 元组表示则每个状态为 $(a_1, ..., a_n)$ 其中 $a_1, ..., a_n$ 分别为 $v_1, ..., v_n$ 的值。
迁移集合由 FTS 的迁移产生。 设 $t \in T$ 。 若 $\sigma \xrightarrow{t} \sigma'$ ，则 $((\sigma(v_1), \sigma(v_2), ..., \sigma(v_n)), (\sigma'(v_1), \sigma'(v_2), ..., \sigma'(v_n))) \in \Delta$ 。
初始状态集合 I 根据 Θ 定义产生。若 $\sigma \models \Theta$ 为真，则 $(\sigma(v_1), \sigma(v_2), ..., \sigma(v_n)) \in I$ 。

对每个变量 v ，设其值域为 $\{a_1, ..., a_{|v|}\}$ 。我们可以构造 $|v|$ 个命题 $\{p_1, ..., p_{|v|}\}$ ，每个命题代表 $v = a_1, v = a_2, ..., v = a_{|v|}$ 中对应的一项。 设 p 代表 $v_i = a$ ， $p \in L((x_1, ..., x_n))$ 当且仅当 $x_i = a$ 。这样，我们有一个原子命题集合 AP 和其上的标号 Kripke 结构 $\langle S, \Delta, I, L\rangle$ 。

扩展标号 KS

标号 KS 只考虑在每个状态上某些简单命题是否成立，我们可以对其进行扩展，在每个状态上标上某个公式是否成立。

Definition 2.6 给定一个原子命题集合 AP 。一个 AP 上的扩展标号 KS 是一个四元组

$$\langle S, \Delta, I, L \rangle$$

其中 $\langle S, \Delta, I \rangle$ 为 *Kripke* 结构， $L : S \to \mathcal{L}_{AP}$ 为状态集合到命题逻辑公式集合的映射。

给定一个 AP 上的扩展标号 Kripke 结构 $\langle S, \Delta, I, L \rangle$ ，一个状态 $s \in S$ 满足性质 $\varphi \in \mathcal{L}_{AP}$ 记作 $M, s \models \varphi$ 当且仅当 $L(s) \to \varphi$ 。

安全性质

系统满足安全性质 $\varphi \in \mathcal{L}_{AP}$, 当且仅当系统的每个运行中的每个状态都满足 φ , 即 $\forall s \in rh(I).(L(s) \to \varphi)$ 。

系统满足可达性质 $\varphi \in \mathcal{L}_{AP}$, 当且仅当存在可以满足 φ 的可达状态, 即 $\exists s \in rh(I).(L(s) \wedge \varphi)$ 可满足。

§2.4.3 公平 Kripke 结构

对于 Kripke 结构而言，只要满足状态转换关系的无穷状态序列就是系统的一个运行。有时候为了更精确地描述系统的运行，我们需要对运行做一定限制以排除一些满足状态转换关系但不合理的无穷状态序列。 若要求能够排除这样的无穷状态序列，我们需要在结构中加入新的成分。我们在 Kripke 结构的基础上，为其运行设置可接受条件，定义公平 Kripke 结构。

Definition 2.7 一个公平 *Kripke* 结构是一个四元组

$$\langle S, \Delta, I, F \rangle$$

其中 $\langle S, \Delta, I \rangle$ 是一个 *Kripke* 结构， $F \subseteq 2^S$ 为公平 *Kripke* 结构的接受状态集的集合。

运行

公平 Kripke 结构 $\langle S, \Delta, I, F \rangle$ 上的一个运行即 Kripke 结构 $\langle S, \Delta, I \rangle$ 上的一个运行。一个公平 Kripke 结构上的运行 $\pi = s_0 s_1 s_2 ...$ 是公平的，当且仅当对所有 $f \in F$ ，

$$inf(\pi) \cap f \neq \emptyset$$

§2.5 标号迁移系统与自动机

Kripke 结构描述的主要是状态和状态之间的转换关系，并不关心是什么动作使得状态发生了变化。在某些时候动作也是系统描述中关键的一部分。 比如自动售货机，按一个键，就是一个动作，按不同的键，就是不同的动作。描述自动售货机的操作过程，就需要动作的描述。

标号迁移系统 (LTS)

Definition 2.8 一个标号迁移系统是一个四元组

$$\langle \Sigma, S, \Delta, I \rangle$$

其中 Σ 为标号集合， S 为状态集合， $\Delta \subseteq S \times \Sigma \times S$ 是一个有标号的迁移关系， I 为初始状态集。

运行

我们用 $s \xrightarrow{a} s'$ 表示存在从 s 到 s' 的 a 迁移，即 $(s, a, s') \in \Delta$ 。 LTS 上的一个运行是状态集 S 上的一个无穷序列

$$s_0 s_1 s_2 ...$$

其中 $s_0 \in I$ 且对任意 $i \geq 0$, 存在 $a \in \Sigma$ ，使得 $s_i \xrightarrow{a} s_{i+1}$ 。

字符串

LTS 上的一个字符串是状态集 Σ 上的一个无穷序列

$$\omega = a_1 a_2 a_3 ...$$

满足条件：存在 LTS 上的一个运行 $\pi = s_0 s_1 s_2 ...$ 使得对任意 $i \geq 0$, $s_i \overset{a_{i+1}}{\rightarrow} s_{i+1}$ 。 π 称为字符串 ω 上的一个运行。

双标号迁移系统

与 KS 类似，我们可以在 LTS 的状态上记载一些信息。

Definition 2.9 *给定一个原子命题集合 AP 。一个 AP 上的标号迁移结构是一个五元组*

$$\langle \Sigma, S, \Delta, I, L \rangle$$

其中 $\langle \Sigma, S, \Delta, I \rangle$ 是一个标号迁移系统， $L : S \to 2^{AP}$ 为状态集到 AP 的幂集的映射。

ω 自动机

与公平 Kripke 结构类似，在 LTS 的基础上，为其运行设置可接受条件，我们就得到 ω 自动机。

§2.5.1 Büchi 自动机

为简单起见，我们先定义只有一个简单接受条件的 Büchi 自动机。

Definition 2.10 *一个 Büchi 自动机是一个五元组*

$$\langle \Sigma, S, \Delta, I, F \rangle$$

其中 $\langle \Sigma, S, \Delta, I \rangle$ 是一个标号迁移系统， $F \subseteq S$ 为自动机的接受状态集。

可接受运行

设 $\mathcal{B} = \langle \Sigma, S, \Delta, I, F \rangle$ 是 Büchi 自动机。 $\mathcal{B}$ 上的运行就是标号迁移系统 $\langle \Sigma, S, \Delta, I \rangle$ 上的运行。定义 $inf(\pi)$ 为无限多次出现在 π 中的状态的集合。 $\mathcal{B}$ 上的运行

$$\pi = s_0 s_1 s_2 \cdots$$

是可接受的当且仅当

$$inf(\pi) \cap F \neq \emptyset$$

可接受字符串

$\mathcal{B}$ 的标号集 Σ 上的一个字符串 $\omega = a_0 a_1 a_2 \cdots$ 是可接受的当且仅当存在 ω 上的可接受运行。 $\mathcal{B}$ 上可接受字符串的集合为 $\mathcal{B}$ 的语言，记作 $\mathcal{L}(\mathcal{B})$ 。

非空性分析算法

给定一个 ω 自动机，其语言是否为空可以先计算 ω 自动机的迁移关系的强连通分图。 如果每个由初始状态可达的非平凡强连通分图（至少有一条边）的顶点的集合与自动机的接受状态的集合都没有共同元素，则该语言为空。为了更好的计算效率，以下算法不先计算所有强连通分图，而是逐个分析。 用 accept(q) 表示 $q \in F$ 是接受状态。该算法由两个嵌套的深度优先算法来实现。

```
start()
{
    W = A = B = {};
    for each initial state s ∈ I, if (s is not in A) { add s to W; dfs1(); }
}
```

```
dfs1()
{
    q = last element from W;
    add q to A;
    for each successor state s of q if (s is not in A) { add s to W; dfs1(); }
    if (accept(q)) { add q to B; dfs2(); }
    delete q from W;
}
```

```
dfs2()
{
    q = last element from B;
    for each successor state s of q
        if (s is in W) report("nonempty");
        if (s is not in B) { add s to B; dfs2(); }
}
```

算法性质

(1) 若 q 不在任何强连通图，则 q 后的所有状态先于 q 处理完毕。
(2) 给定一个 Büchi 自动机，算法报告 "nonempty" 当且仅当该自动机的语言非空。
(3) 给定一个 Büchi 自动机，则该自动机的非空性判定是线性的。

§2.5.2 扩展 Büchi 自动机

为了能够表示多个接受条件，我们扩展 Büchi 自动机的定义，将接受状态集合改为接受状态集的集合。

Definition 2.11 *一个扩展 Büchi 自动机是一个五元组*

$$\langle \Sigma, S, \Delta, I, F \rangle$$

其中 $\langle \Sigma, S, \Delta, I \rangle$ 是一个标号迁移系统， $F \subseteq 2^S$ 为自动机的接受状态集的集合。

可接受运行

设 $\mathcal{B} = \langle \Sigma, S, \Delta, I, F \rangle$ 是扩展 Büchi 自动机。 $\mathcal{B}$ 上的运行就是标号迁移系统 $\langle \Sigma, S, \Delta, I \rangle$ 上的运行。 $\mathcal{B}$ 上的运行

$$\pi = s_0 s_1 s_2 \cdots$$

是可接受的，当且仅当对所有 $f \in F$ ，

$$inf(\pi) \cap f \neq \emptyset$$

扩展 Büchi 自动机的表达能力

从语言的角度讲，扩展 Büchi 自动机的表达能力和 Büchi 自动机是一样的。给定扩展 Büchi 自动机 $\mathcal{B} = \langle \Sigma, S, \Delta, I, \{f_1, ..., f_n\} \rangle$ 。定义

$n(s,i)$	$=$	if $(i = n)$ then 0; else if $(s \in f_{i+1})$ then $i+1$; else i.
S'	$=$	$S \times \{0, ..., n\}$
Δ'	$=$	$\{(s,i), a, (s', n(s',i)) \mid (s, a, s') \in \Delta, i \in \{0, ..., n\}\}$
I'	$=$	$I \times \{0\}$
F'	$=$	$S \times \{n\}$

定义 Büchi 自动机 $\mathcal{B}' = \langle \Sigma, S', \Delta', I', F' \rangle$ ，则 $\mathcal{L}(\mathcal{B}') = \mathcal{L}(\mathcal{B})$ 。

自动机的运算

自动机的运算包括并、交和补。以下定义扩展 Büchi 自动机的并和交。

给定两个扩展 Büchi 自动机 $\mathcal{B}_1 = \langle \Sigma, S_1, \Delta_1, I_1, F_1 \rangle, \mathcal{B}_2 = \langle \Sigma, S_2, \Delta_2, I_2, F_2 \rangle$ 且 $S_1 \cap S_2 = \emptyset$ 。设

$$F = \{f \cup S_2 \mid f \in F_1\} \cup \{f \cup S_1 \mid f \in F_2\}$$

定义 $\mathcal{B}_1 \cup \mathcal{B}_2 = \langle \Sigma, S_1 \cup S_2, \Delta_1 \cup \Delta_2, I_1 \cup I_2, F \rangle$ ，则 $\mathcal{L}(\mathcal{B}_1 \cup \mathcal{B}_2) = \mathcal{L}(\mathcal{B}_1) \cup \mathcal{L}(\mathcal{B}_2)$ 。

给定两个扩展 Büchi 自动机 $\mathcal{B}_1 = \langle \Sigma, S_1, \Delta_1, I_1, F_1 \rangle, \mathcal{B}_2 = \langle \Sigma, S_2, \Delta_2, I_2, F_2 \rangle$ 且 $S_1 \cap S_2 = \emptyset$ 。设

$$\Delta = \{((q_1, q_2), a, (q_1', q_2')) \mid (q_1, a, q_1') \in \Delta_1, (q_2, a, q_2') \in \Delta_2\}$$
$$F = \{f \times S_2 \mid f \in F_1\} \cup \{S_1 \times f \mid f \in F_2\}$$

定义 $\mathcal{B}_1 \cap \mathcal{B}_2 = \langle \Sigma, S_1 \times S_2, \Delta, I_1 \times I_2, F \rangle$ ，则 $\mathcal{L}(\mathcal{B}_1 \cap \mathcal{B}_2) = \mathcal{L}(\mathcal{B}_1) \cap \mathcal{L}(\mathcal{B}_2)$ 。

§2.5.3 Streett 自动机和 Muller 自动机

扩展 Büchi 自动机是 Büchi 自动机的一种扩展，在扩展 Büchi 自动机基础上再对其接受条件做进一步扩展或重新定义，我们可以得到 Streett 自动机或 Muller 自动机。

Definition 2.12 一个 *Streett* 自动机是一个五元组

$$\langle \Sigma, S, \Delta, I, F \rangle$$

其中 $\langle \Sigma, S, \Delta, I \rangle$ 是一个标号迁移系统， $F \subseteq 2^S \times 2^S$ 为自动机的接受状态集的集合。

可接受运行

设 $\mathcal{B} = \langle \Sigma, S, \Delta, I, F \rangle$ 是 Streett 自动机。$\mathcal{B}$ 上的运行就是标号迁移系统 $\langle \Sigma, S, \Delta, I \rangle$ 上的运行。$\mathcal{B}$ 上的运行

$$\pi = s_0 s_1 s_2 \cdots$$

是可接受的，当且仅当对所有 $(f, g) \in F$ ，

$$inf(\pi) \cap f \neq \emptyset \rightarrow inf(\pi) \cap g \neq \emptyset$$

Definition 2.13 一个 *Muller* 自动机是一个五元组

$$\langle \Sigma, S, \Delta, I, F \rangle$$

其中 $\langle \Sigma, S, \Delta, I \rangle$ 为迁移系统， $F \subseteq 2^S$ 为自动机的接受状态集的集合。

可接受运行

设 $\mathcal{B} = \langle \Sigma, S, \Delta, I, F \rangle$ 是 Muller 自动机。$\mathcal{B}$ 上的运行就是标号迁移系统 $\langle \Sigma, S, \Delta, I \rangle$ 上的运行。$\mathcal{B}$ 上的运行

$$\pi = s_0 s_1 s_2 \cdots$$

是可接受的，当且仅当

$$inf(\pi) \in F$$

表达能力

从语言的角度讲， Büchi 自动机、扩展 Büchi 自动机、 Streett 自动机与 Muller 自动机的表达能力相同，且 Muller 自动机是补封闭的。

§2.5.4 基于迁移的扩展 Büchi 自动机

以上自动机的接受条件都是状态集合。我们也可以将接受条件定义在迁移上。由于基于迁移的 Büchi 自动机是基于迁移的扩展 Büchi 自动机的一个特例，我们直接定义基于迁移的扩展 Büchi 自动机。

Definition 2.14 基于迁移的扩展 *Büchi* 自动机是一个五元组

$$\langle S, \Sigma, \Delta, I, F \rangle$$

其中 $\langle \Sigma, S, \Delta, I \rangle$ 是一个标号迁移系统， $F \subseteq 2^{\Delta}$ 是接受迁移集合。

可接受运行

设 $\mathcal{B} = \langle \Sigma, S, \Delta, I, F \rangle$ 是基于迁移的 Büchi 自动机。$\mathcal{B}$ 上的运行就是标号迁移系统 $\langle \Sigma, S, \Delta, I \rangle$ 上的运行。给定一个 $\mathcal{B}$ 上的运行 $\pi = s_0 s_1 s_2 \cdots$ ，一个迁移 (s, a, s') 可能出现在该路径中当且仅当存在 $i \geq 0$ 使得 $s = s_i$ 且 $s' = s_{i+1}$ 。定义 $inf_{\Delta}(\pi)$ 为可能无限多次出现在路径 π 中的迁移的集合。$\mathcal{B}$ 上的运行

$$\pi = s_0 s_1 s_2 \cdots$$

是可接受的，当且仅当对所有 $f \in F$ ，

$$inf_{\Delta}(\pi) \cap f \neq \emptyset$$

表达能力

从语言的角度讲，基于迁移的扩展 Büchi 自动机的表达能力和 Büchi 自动机是一样的。给定基于迁移的扩展 Büchi 自动机 $\mathcal{B} = \langle \Sigma, S, \Delta, I, F \rangle$ ，定义

$n(s, a, s', i)$	$=$	if $(i = n)$ then 0; else if $((s, a, s') \in f_{i+1})$ then $i + 1$; else i.
S'	$=$	$S \times \{0, ..., n\}$
Δ'	$=$	$\{(s, i), a, (s', n(s, a, s', i)) \mid (s, a, s') \in \Delta, i \in \{0, ..., n\}\}$
I'	$=$	$I \times \{0\}$
F'	$=$	$S \times \{n\}$

定义 Büchi 自动机 $\mathcal{B}' = \langle \Sigma, S', \Delta', I', F' \rangle$ ，则 $\mathcal{L}(\mathcal{B}') = \mathcal{L}(\mathcal{B})$ 。

§2.5.5 确定型自动机

ω- 自动机由迁移系统和接受条件组成。不同的接受条件定义了不同的自动机。我们也可以对迁移系统分类，得到确定型自动机和非确定型自动机。确定型自动机的初始状态只有一个且其迁移关系受以下限制：

$$(s, a, s'), (s, a, s'') \in \Delta \Rightarrow s' = s''$$

给定一个字符串。对确定型自动机来讲，若有与之匹配的运行，则这样的运行是唯一的。

表达能力

确定型 Muller 自动机是补封闭的，且其表达能力与非确定型 Muller 自动机相同。确定型 Büchi 自动机不是补封闭的。

定义 $\mathcal{B} = \langle \Sigma, S, \Delta, I, F \rangle$ 其中

$\Sigma = \{a, b\}$
$S = \{s_0, s_1\}$
$\Delta = \{(s_0, a, s_1), (s_0, b, s_0), (s_1, a, s_1), (s_1, b, s_0)\}$
$I = \{s_0\}$
$F = \{s_0\}$

定义 $\mathcal{B}' = \langle \Sigma, S, \Delta, I, F \rangle$ 其中

$\Sigma = \{a, b\}$
$S = \{s_0, s_1\}$
$\Delta = \{(s_0, a, s_0), (s_0, b, s_0), (s_0, a, s_1), (s_1, a, s_1)\}$
$I = \{s_0\}$
$F = \{s_1\}$

则 $\mathcal{B}$ 是一个确定型 Büchi 自动机，而 $\mathcal{B}'$ 是一个非确定型 Büchi 自动机。$\mathcal{B}'$ 的语言 $\mathcal{L}(\mathcal{B}') = (a + b)^* a^\omega$ 是 $\mathcal{L}(\mathcal{B}) = (a^* b)^\omega$ 的补，但 $\mathcal{L}(\mathcal{B}')$ 不是确定型 Büchi 自动机可以表达的。

§2.6 标号交错迁移系统与交错自动机

标号迁移系统的每个动作其后续状态可能有多个，但每次运行只能从其中选择一个状态。对这类迁移系统的一种扩展是让其可以选择多个后续状态。这样扩展的迁移系统的一次运行可以是树状结构。

标号交错迁移系统 (ATS)

Definition 2.15 一个标号交错迁移系统是一个四元组 $\langle \Sigma, S, \Delta, I \rangle$，其中 Σ 为动作集合，S 为状态集合，$\Delta \subseteq S \times \Sigma \times 2^S$ 为标号的迁移关系，I 为系统初始状态的集合。

字符串上的运行

给定一个 Σ 上的无限长的字符串 $\omega = a_0 a_1 a_2 ...$。标号交错迁移系统在 ω 上的一个运行是状态集 S 上的一颗树 r。设 $r(0)$ 为 r 的根节点的单点集、$r(i)$ 为 r 的第 i 层节点的集合、$child(x)$ 为节点 x 的子节点的集合。r 满足以下条件。

$r(0) \subseteq I$
若 $x \in r(i)$ 为第 i 层的一个节点，则 $child(x) \in \Delta(x, a_i)$

若 $\Delta(x, a_i)$ 包含空集，则 x 可以没有子节点。

运行

状态集 S 上的一颗树 r 是标号交错迁移系统 $\langle \Sigma, S, \Delta, I \rangle$ 的一个运行当且仅当存在一个 Σ 上的无限长的字符串 $\omega = a_0 a_1 a_2 ...$ 使得 r 是标号交错迁移系统在 ω 上的一个运行。

双标号交错迁移系统

与 KS 类似，我们可以在 ATS 的状态上记载一些信息。

Definition 2.16 给定一个原子命题集合 AP。一个 AP 上的双标号交错迁移系统是一个五元组

$$\langle \Sigma, S, \Delta, I, L \rangle$$

其中 $\langle \Sigma, S, \Delta, I \rangle$ 是一个标号交错迁移系统，$L : S \to 2^{AP}$ 为状态集到 AP 的幂集的映射。

交错 Büchi 自动机

Definition 2.17 一个交错 $Büchi$ 自动机是一个五元组

$$\langle \Sigma, S, \Delta, I, F \rangle$$

其中 $\langle \Sigma, S, \Delta, I \rangle$ 是一个标号交错迁移系统，$F \subseteq S$ 为自动机的接受状态集。

可接受运行

设 $\mathcal{B} = \langle \Sigma, S, \Delta, I, F \rangle$ 是一个交错 Büchi 自动机。$\mathcal{B}$ 上的运行就是标号交错迁移系统 $\langle \Sigma, S, \Delta, I \rangle$ 上的运行。$\mathcal{B}$ 的一个运行树 r 是可接受的，当且仅当 r 的所有无穷路径 ρ 满足

$$inf(\rho) \cap F \neq \emptyset$$

表达能力

$\mathcal{B}$ 的字母表 Σ 上的一个无限长字符串

$$\omega = a_0 a_1 a_2 \cdots$$

对 $\mathcal{B}$ 来说是可接受的当且仅当存在 ω 上的可接受运行树。交错 Büchi 自动机与 Büchi 自动机的表达能力相同。

§2.7 时间自动机

为了描述迁移动作之间时间长短的限制，我们需要在模型中加入时钟的概念。时间的限制由时间变量上的公式表示。设 X 为时钟变量的集合， Q 为时间常量的集合。时钟公式的集合 $\Phi(X)$ 由以下语法给出。

$$\phi ::= x \leq c \mid c \leq x \mid \neg\phi \mid \phi \wedge \phi \mid$$

其中 $x \in X$ 为时钟变量， $c \in Q$ 为时间常量。一个时钟赋值 v 是一个 X 到 R 的函数。我们用 $v + t$ 表示对所有 $x \in X$ 满足 $v'(x) = v(x) + t$ 的时钟赋值 v' ； 用 $t \cdot v$ 表示对所有 $x \in X$ 满足 $v'(x) = t \cdot v(x)$ 的时钟赋值 v' ； 用 $[Y \to t]v$ 表示对所有 $x \in X \setminus Y$ 满足 $v'(x) = v(x)$ 和对所有 $x \in Y$ 满足 $v'(x) = t$ 的时钟赋值 v' 。

时间迁移系统 (TTS)

Definition 2.18 一个时间迁移系统是一个五元组

$$\langle \Sigma, S, C, \Delta, I \rangle$$

其中 Σ 为字母表， S 为状态集合， C 为时钟变量集合， $\Delta \subseteq S \times \Sigma \times 2^C \times \Phi(C) \times S$ 为迁移关系， $I \in S$ 为系统初始状态的集合。

时间字符串上的运行

设 $V = C \to R$ 为时钟赋值的集合。定义 $\tau_0 = 0$ 。时间迁移系统 $\langle \Sigma, S, C, \Delta, I \rangle$ 在时间字符串 $(\sigma, \tau) = (\sigma_1, \tau_1)(\sigma_2, \tau_2) \cdots \in (\Sigma \times R)^\omega$ 上的一个运行是状态集 $S \times V$ 上的一个无穷序列

$$(s_0, v_0)(s_1, v_1)(s_2, v_2)...$$

其中

$s_0 \in I$,
对所有 $x \in C$,$v_0(x) = 0$,
对任意 $i \geq 0$ ，存在 $\lambda \subseteq C, \varphi \in \Phi(C)$ 使得
$(s_i, \sigma_{i+1}, \lambda, \varphi, s_{i+1}) \in \Delta$ 且 $(v_i + \tau_{i+1} - \tau_i)$ 满足 φ, $v_{i+1} = [\lambda \to 0](v_i + \tau_{i+1} - \tau_i)$ 。

运行

状态集 $S \times V$ 上的一个无穷序列 r 是时间迁移系统 $\langle \Sigma, S, C, \Delta, I \rangle$ 一个运行当且仅当存在一个时间字符串 (σ, τ) 使得 r 是时间迁移系统在 (σ, τ) 上的一个运行。

时间 Büchi 自动机

Definition 2.19 一个时间 *Büchi* 自动机是一个六元组

$$\langle \Sigma, S, C, \Delta, I, F \rangle$$

其中 $\langle \Sigma, S, C, \Delta, I \rangle$ 构成一个时间迁移系统， $F \subseteq S$ 为自动机的接受状态集。

可接受运行

时间 Büchi 自动机 $\mathcal{B} = \langle \Sigma, S, C, \Delta, I, F \rangle$ 。一个时间字符串 (σ, τ) 上的 B 的运行 $r = (s, v)$ 是可接受的，当且仅当 $inf(r) \cap F \neq \emptyset$ ，其中 $inf(r)$ 代表无限多次出现在运行 r 中的状态的集合。

语言

若一个时间字符串 (σ, τ) 上存在 B 的可接受运行，则称 (σ, τ) 是可接受。 $\mathcal{B}$ 上可接受时间字符串的集合为 $\mathcal{B}$ 的语言，记作 $\mathcal{L}(\mathcal{B})$ 。

§2.8 Petri 网

Petri 网的特点是其可以描述真并发。其基本要素有库所、迁移、连接库所和迁移的有向边。由库所进入迁移的边可以看成是输入条件，由迁移进入库所的边可以看成是迁移的输出。

Definition 2.20 一个 *Petri* 网是一个四元组

$$\langle P, T, F, M_0 \rangle$$

其中 P 为位置集合， T 为迁移集合， $F \subseteq (P \times T) \cup (T \times P)$ 为边的集合， $M_0 : P \to N$ 为初始状态，其中 N 为自然数集合。

迁移

定义 ${}^\circ p\ (t) = \{p \in P \mid (p, t) \in F\}$ 和 $p^\circ\ (t) = \{p \in P \mid (t, p) \in F\}$ 。迁移 $t \in T$ 在状态 M 是可执行的，当且仅当输入条件满足，即 $\forall p \in {}^\circ p\ (t), M(p) \geq 1$ 。 我们用 $M \xrightarrow{t} M'$ 表示 $t \in T$ 在状态 M 是可执行且 $\forall p \in P$ ， $M'(p) = M(p) - \alpha_0(p, t) + \alpha_1(p, t)$ 其中

$\alpha_0(p, t) = 1$ 当且仅当 $p \in {}^\circ p\ (t)$
$\alpha_1(p, t) = 1$ 当且仅当 $p \in p^\circ\ (t)$

可达性

给定一个状态 M 。该状态是否可达是可判定的且是 EXPSPACE 难的。一个 Petri 网 $\langle P, T, F, M_0 \rangle$ 是 k 界的，当且仅当对其所有可达状态 M 有 $\forall p \in P. M(p) \leq k$ 。 1 界 Petri 网称为安全 Petri 网。对于安全 Petri 网，状态可达的判定是 PSPACE 完备的。

§2.9 通信系统

通信系统模型用来表达多个进程通过通道交换信息进行控制和计算的过程。每个进程内部有状态和状态迁移。状态迁移的原因可以是内部事件或输入输出。

§2.9.1 通道

Definition 2.21 给定一个类型 $\langle S, N\rangle$ 其中 S 为字母表的， N 为自然数。一个类型为 $\langle S, N\rangle$ 的通道 m ，记作 $m \in \langle S, N\rangle$ ，是一个值域为 $\bigcup_{i=0}^{N}\{\langle a_1, ..., a_i\rangle \,|\, a_i \in S\}$ 的变量。

通道状态记载通道的给定赋值，记为 σ 。若 $m \in \langle S, N\rangle$ ，则 $\sigma(m) \in \bigcup_{i=0}^{N}\{\langle a_1, ..., a_i\rangle \,|\, a_i \in S\}$ 。通道状态空间为通道状态的集合，记为 Σ 。

事件

若 $m \in \langle S, N\rangle$ ，则与 m 相关的事件集合为 $ACT(m) = \{m?s \,|\, s \in S\} \cup \{m!s \,|\, s \in S\} \cup \{\epsilon\}$ 。设 $C = \{m_1, ..., m_n\}$ 为通道集合。则与 C 相关的事件集合为 $ACT(C) = ACT(m_1) \cup \cdots \cup ACT(m_n)$ 。与通道无关的内部事件记为 $\{\epsilon\}$ 。

可执行事件

对于 $a = \langle a_1, ..., a_n\rangle$ ，定义 $|a| = n$, $a \vdash s = \langle a_1, ..., a_n, s\rangle$, $HEAD(a) = a_1$, $TAIL(a) = \langle a_2, ..., a_n\rangle$ 。给定一个通道状态 σ 和一个通道 $m \in \langle S, N\rangle$ 。若以下一项成立则 $a \in ACT(m) \cup \{\epsilon\}$ 可执行。

$a = \epsilon$
$a = m!s$ 且 $\lvert\sigma(m)\rvert < N$ 且 $s \in S$
$a = m?s$ 且 $\lvert\sigma(m)\rvert > 0$ 且 $s = HEAD(\sigma(m))$

给定 $\sigma \in \Sigma$,$m \in \langle S, N\rangle$,$a \in ACT(m)$ 。我们用 $\sigma \xrightarrow{t} \sigma'$ 表示 a 在 σ 可执行且

若 $a = \epsilon$,	则 $\sigma' = \sigma$
若 $a = m!s$,	则 $\sigma' = \sigma[m/\sigma(m) \vdash s]$
若 $a = m?s$,	则 $\sigma' = \sigma[m/TAIL(\sigma(m))]$

§2.9.2 通信单元

Definition 2.22 一个通信单元是一个四元组

$$\langle Q, C, \Delta, q_0\rangle$$

其中 Q 为状态集合， C 为通道集合， $\Delta \subseteq Q \times ACT(C) \times Q$ 为标号的迁移关系， $q_0 \in Q$ 为初始状态。

状态

给定一个通信单元 $\langle Q, C, \Delta, q_0\rangle$ ，状态由两部分组成 (s, σ) 其中 $s \in Q$ 为控制状态， $\sigma \in \Sigma$ 为通道状态。

运行

迁移 (q,a,q') 在 (s,σ) 可执行，当且仅当 $q=s$ 且 a 在 σ 可执行。设 $C=\{m_1,...,m_k\}$。一个运行是 $Q\times\Sigma$ 上的一个无穷序列

$$(z_0,\sigma_0)(z_1,\sigma_1)(z_2,\sigma_2)...$$

其中 $z_0=q_0$, $\sigma_0(m_1)=\cdots=\sigma_0(m_k)=\langle\rangle$ 且对任意 $i\geq 0$,

$$\frac{\begin{array}{l}\text{存在 } (q,a,q')\in\Delta\\ (q,a,q') \text{ 在 } (z_i,\sigma_i) \text{ 可执行}\end{array}}{z_{i+1}=q' \text{ 且 } \sigma_i\xrightarrow{a}\sigma_{i+1}}$$

§2.9.3 通信系统

Definition 2.23 一个通信系统 $\{P_1,...,P_n\}$ 是由通信单元 $P_1=\langle Q_1,C_1,\Delta_1,q_{10}\rangle$, ..., $P_n=\langle Q_n,C_n,\Delta_n,q_{n0}\rangle$ 组成的集合，其中 $Q_1,...,Q_n$ 为两两不相交的集合。

状态

定义 $S_1\otimes\cdots\otimes S_n=\{\{s_1,...,s_n\}\mid s_1\in S_1,...,s_n\in S_n\}$。系统状态的集合为

$$(Q_1\otimes\cdots\otimes Q_n)\times\Sigma$$

运行

一个迁移 $(q,a,q')\in\Delta_i$ 在状态 (Q,σ) 可执行当且仅当 $q\in Q$ 且 a 在 σ 可执行。设 $C_1\cup\cdots\cup C_n=\{m_1,...,m_k\}$。一个运行是 $(Q_1\otimes\cdots\otimes Q_n)\times\Sigma$ 上的一个无穷序列

$$(z_0,\sigma_0)(z_1,\sigma_1)(z_2,\sigma_2)...$$

其中 $z_0=\{q_0,...,q_n\}$, $\sigma_0(m_1)=\cdots=\sigma_0(m_k)=\langle\rangle$ 且对任意 $i\geq 0$,

$$\frac{\begin{array}{l}\text{存在 } (q,a,q')\in\Delta\\ (q,a,q') \text{ 在 } (z_i,\sigma_i) \text{ 可执行}\end{array}}{z_{i+1}=(z_i\setminus\{q\})\cup\{q'\} \text{ 且 } \sigma_i\xrightarrow{a}\sigma_{i+1}}$$

并发算子

给定两个通信单元 $P_1=\langle Q_1,C_1,\Delta_1,q_{10}\rangle$, $P_2=\langle Q_2,C_2,\Delta_2,q_{20}\rangle$ 且 $Q_1\cap Q_2=\emptyset$。其并 $P_1||P_2$ 为

$$\langle Q_1\otimes Q_2,C_1\cup C_2,\Delta,\{q_{10},q_{20}\}\rangle$$

其中

$$\Delta=\{(\{q_1,q_2\},a,\{q_1',q_2\})\mid(q_1,a,q_1')\in\Delta_1\}\cup\{(\{q_1,q_2\},a,\{q_1,q_2'\})\mid(q_2,a,q_2')\in\Delta_2\}$$

并发算子满足结合率和交换率。

有穷值域变量

设 m 的值域为 $\{0,1,2,...,n\}$。用 $[n]$ 表示 $\{0,1,2,...,n\}$。则 m 可以表示为 $B=\langle Q,M,\Delta,q_0\rangle$，其中

$$
\begin{array}{l}
Q=\{q_0,q_1,q_2,...,q_n,r_0,r_1,r_2,...,r_n\}\\
M=\{mi,mo\mid mi,mo\in\langle\{0,1,2,...,n,r\},1\rangle\}\\
\Delta=\{(q_i,mi?j,q_j)\mid i,j\in[n]\}\cup\{(q_i,mi?r,r_i)\mid i\in[n]\}\cup\{(r_i,mo?i,q_i)\mid i\in[n]\}
\end{array}
$$

设 m_1,m_2 为两个值域为 $\{0,1,2,...,n\}$ 的变量，$c\in\langle\{0,1,2,...,n\},k\rangle$ 为通道。则以下左边的迁移可用右边的迁移集合实现。

$(q,c?m_1,q')$	$\{(q,c?i,q_{1i})\mid i\in[n]\}\cup\{(q_{1i},m_1i!i,q')\mid i\in[n]\}$
$(q,c!m_1,q')$	$\{(q,m_1i!r,q_1)\}\cup\{(q_1,m_1o?i,q_{1i})\mid i\in[n]\}\cup\{(q_{1i},c!i,q')\mid i\in[n]\}$
$(q,m_1=i,q')$	$\{(q,m_1i!i,q')\}$
$(q,m_1=m_2,q')$	$\{(q,m_2i!r,q_1)\}\cup\{(q_1,m_2o?i,q_{1i})\mid i\in[n]\}\cup\{(q_{1i},m_1i!i,q')\mid i\in[n]\}$

同样，我们可以实现变量的比较和算术运算。因此在通信系统上增加有穷值域变量的使用，并不增强通信系统的表达能力。

§2.10 说明

本章主要内容为程序及系统运行的模型。有关一阶迁移系统部分可参考 [Pel01] 。 Kripke 结构和有限自动机可参考 [CGP99] 。 Petri 网的介绍可参考 [NW96] 。交错自动机部分可参考 [Var97, AHK97] 。时间自动机部分可参考 [AD94] 。可达性算法、自动机非空性算法和通信系统可参考 [Hol90] 。

参考文献

[AD94] Rajeev Alur and David L. Dill. A Theory of Timed Automata. Theor. Comput. Sci. 126(2): 183-235 (1994).

[AHK97] Rajeev Alur, Thomas A. Henzinger and Orna Kupferman. Alternating-Time Temporal Logic. COMPOS 1997: 23-60.

[Hol90] Gerard J. Holzmann. Design and Validation of Computer Protocols. Prentice Hall, 1990.

[CGP99] E. Clark, O. Grumberg and D. Peled. Model Checking. MIT press, 1999.

[Pel01] Doron A. Peled. Software Reliability Methods. Springer-Verlag. 2001.

[NW96] Mogens Nielsen and Glynn Winskel. Petri Nets and Bisimulation. Theor. Comput. Sci. 153(1&2): 211-244 (1996).

[Var97] Moshe Y. Vardi. Alternating Automata: Unifying Truth and Validity Checking for Temporal Logics. CADE 1997: 191-206.