.

软件系统行为与程序正确性
— 迁移系统分析

前言

迁移系统是描述具有状态变化的动态系统的数学模型。迁移系统分析的目标是通过分析确认迁移系统是否具备给定的特性，如安全性质和必达性质等。

本书介绍迁移系统及其性质的描述与分析方法。本书的内容以离散迁移系统为主。相关性质的描述方法包括：用于描述并发系统的时序逻辑和用于描述顺序计算部分正确性和终止性的程序逻辑。

本书第 1 章介绍预备知识，第 2-4 章介绍不同类型的迁移系统，第 5-6 章介绍时序逻辑，第 7-8 章分别介绍推理验证和模型检测方法，第 9 章介绍实例分析。附录 A 介绍第 9 章的实例分析中用到的模块化迁移系统建模语言。

本书第 5.1.2 节和 8.3.1 节的内容主要基于作者的研究工作，第 9 章实例分析中用的模型检测工具 VERDS 是作者实现的软件工具原型，该工具原型实现的内容包括建立在第 5.1.2 节和第 8.3.1 节上的限界正确性检查算法以及其底层的命题逻辑可满足性判定和量词命题逻辑可满足性判定算法。

目 录

§1 预备知识 **1**

§1.1 命题逻辑 . . . 1
§1.2 谓词逻辑 . . . 2
§1.3 集合 . . . 4
§1.4 关系 . . . 5
§1.5 函数 . . . 7
§1.6 完全偏序和格上的不动点 . . . 8
§1.7 有向图 . . . 9
§1.8 练习 . . . 9

§2 状态迁移模型 **10**

§2.1 Kripke 模型 . . . 10
§2.1.1 安全性质与可达性质的分析 . . . 11
§2.1.2 必达性质与可免性质的分析 . . . 12
§2.2 公平 Kripke 模型 . . . 14
§2.2.1 公平安全性质与公平可达性质分析 . . . 15
§2.2.2 公平必达性质与公平可免性质的分析 . . . 16
§2.2.3 模型非空问题 . . . 17
§2.2.4 强公平与弱公平条件 . . . 18
§2.3 标号 Kripke 模型 . . . 19
§2.4 公平标号 Kripke 模型 . . . 21
§2.5 练习 . . . 21

§3 变量迁移模型 **22**

§3.1 卫式迁移模型 . . . 22
§3.1.1 正确性性质 . . . 23
§3.1.2 卫式迁移模型与标号 Kripke 模型的等价 . . . 23
§3.2 公平卫式迁移模型 . . . 24
§3.2.1 正确性性质 . . . 25
§3.2.2 公平卫式迁移模型与公平标号 Kripke 模型的等价 . . . 25
§3.3 流程图模型 . . . 26
§3.3.1 正确性问题 . . . 27
§3.3.2 流程图模型与卫式迁移模型的等价 . . . 28
§3.4 结构化程序模型 . . . 29
§3.4.1 正确性问题 . . . 30
§3.4.2 结构化程序模型与流程图模型的等价 . . . 31
§3.5 练习 . . . 31

§4 标号迁移模型 **32**

§4.1 标号迁移系统 . . . 32
§4.2 无穷字符串上的自动机 . . . 32
§4.2.1 自动机的可接受运行条件的不同设置 . . . 34
§4.2.2 确定型自动机与非确定型自动机 . . . 35

§4.2.3 自动机与 Kripke 模型 . . . 35
§4.3 时间迁移系统与时间自动机 . . . 36
§4.4 混成迁移系统 . . . 37
§4.5 Petri 网模型 . . . 38
§4.6 通信系统 . . . 40
§4.6.1 通道 . . . 40
§4.6.2 通信单元 . . . 40
§4.6.3 通信系统 . . . 40
§4.6.4 通信系统与通信单元 . . . 41
§4.6.5 通信单元与卫式迁移模型的等价 . . . 41
§4.7 练习 . . . 42

§5 线性时序逻辑 43
§5.1 命题线性时序逻辑 (PLTL) . . . 43
§5.1.1 PLTL 公式的推理 . . . 45
§5.1.2 PLTL 限界语义 . . . 45
§5.2 PLTL 公式的不动点表示与线性 μ 演算 (νTL) . . . 47
§5.3 一阶线性时序逻辑 . . . 48
§5.4 练习 . . . 50

§6 分枝时序逻辑 51
§6.1 计算树逻辑 (CTL) . . . 51
§6.1.1 CTL 公式的推理 . . . 52
§6.1.2 CTL 限界语义 . . . 53
§6.2 CTL 公式的不动点表示与模态 μ 演算 . . . 54
§6.2.1 模态 μ 演算 . . . 55
§6.2.2 模态算子的对偶关系与 NNF 范式: . . . 56
§6.2.3 CTL 公式到 μ 演算公式的转换 . . . 56
§6.3 计算树逻辑 CTL* . . . 56
§6.4 练习 . . . 57

§7 推理验证方法 59
§7.1 流程图模型的推理 . . . 59
§7.1.1 直接证明 . . . 59
§7.1.2 基于路径的推理 . . . 59
§7.2 结构化程序模型的推理 . . . 61
§7.2.1 指称语义 . . . 62
§7.2.2 Hoare 逻辑 . . . 63
§7.3 卫式迁移模型的推理 . . . 66
§7.4 练习 . . . 68

§8 模型检测方法 69
§8.1 基于状态的分析 . . . 69
§8.1.1 CTL 性质的状态标号算法 . . . 69
§8.1.2 CTL 性质的不动点算法 . . . 69

§8.1.3 PLTL 性质的基于自动机空性检测的算法 70
§8.2 基于符号模型的分析 70
§8.2.1 符号模型 71
§8.2.2 CTL 性质的符号模型检测 72
§8.2.3 PLTL 性质的符号模型检测 72
§8.3 限界模型检测与限界正确性检查 73
§8.3.1 CTL 公式的限界正确性检查 73
§8.3.2 PLTL 公式的限界模型检测 74
§8.4 练习 75

§9 实例分析 **76**
§9.1 互斥算法 76
§9.1.1 推理验证 76
§9.1.2 基于模型检测算法的验证 78
§9.1.3 基于模型检测工具的自动验证 80
§9.1.4 公平约束模型的模型检测 80
§9.2 整数平方根算法 81
§9.2.1 推理验证 81
§9.2.2 基于模型检测工具的自动验证 83
§9.2.3 错误检查与模型检测中的反例生成 83
§9.3 查找模型中的特定状态和路径 84
§9.4 练习 84

附录 A 建模语言 VML **85**

§1　预备知识

本章介绍逻辑、集合、关系和有向图方面的知识。

§1.1　命题逻辑

命题是具有确定真假意义的陈述句，是逻辑推理的基本元素。真假值用 1 和 0 表示。 简单命题是不可分解的命题。复合命题由简单命题和联结词组成。 通常我们有一元联结词 $\neg$（非）和二元联结词 $\wedge$（合取），$\vee$（析取），$\rightarrow$（蕴涵），$\leftrightarrow$（等价）等。 n- 元联结词可以看成是 $\{0,1\}^n$ 到 $\{0,1\}$ 的函数。我们有 $\neg 1 = 0$ 和 $\neg 0 = 1$ 。 用 A,B,C 等字母表示命题变元，以下是 $\wedge$,$\vee$, $\rightarrow$,$\leftrightarrow$ 的真值表。

A	B		$A \wedge B$	$A \vee B$	$A \rightarrow B$	$A \leftrightarrow B$
0	0		0	0	1	1
0	1		0	1	1	0
1	0		0	1	0	0
1	1		1	1	1	1

命题变元称为原子公式。公式集合由归纳定义生成。 设 S 是联结词的集合。由 S 生成的公式如下。 （1）命题变元（原子公式）是由 S 生成的公式；（2）若 φ 是 S 中的 0 元联结词，则 φ 是由 S 生成的公式；（3）若 f 是 S 中的 n 元 $(n \geq 1)$ 联结词，$\varphi_1, ..., \varphi_n$ 是由 S 生成的公式，则 $f(\varphi_1, ..., \varphi_n)$ 是由 S 生成的公式。 这里用的是前缀记法。对于二元联结词，我们习惯使用中缀记法。我们规定联结词的优先级以省略括号。联结词的优先级按顺序排列如下：$\neg, \wedge, \vee, \rightarrow, \leftrightarrow$ 。

由全体命题变元组成的集合到 $\{0,1\}$ 的函数称为真值赋值。设 v 是真值函数。 记 A^v 为 v 赋给 A 的值。由 S 生成的公式 φ 在 v 下的值定义如下。（1）若 φ 是命题变元 A，则 $v(\varphi) = A^v$；（2）若 φ 是 S 中的 0 元联结词 c，则 $v(\varphi) = c$；（3）若 $\varphi = f(\varphi_1, ..., \varphi_n)$，其中 f 是 S 中的 n 元 $(n \geq 1)$ 联结词，则 $v(\varphi) = f(v(\varphi_1), ..., v(\varphi_n))$ 。

如果真值赋值 v 使得 $v(\varphi) = 1$，则称 v 满足 φ，记作 $v \models \varphi$ 。

如果有真值赋值 v，使得 $v \models \varphi$，则称 φ 为可满足式。否则称 φ 为永假式（不可满足式）。如果对于每个真值赋值 v，都有 $v \models \varphi$，则称 φ 为永真式（重言式）。

如果对于每个真值赋值 v，都有 $v(\varphi) = v(\psi)$，则称 φ 与 ψ 逻辑等价，记作 $\varphi \Leftrightarrow \psi$ 。$\varphi \Leftrightarrow \psi$ 当且仅当 $\varphi \leftrightarrow \psi$ 是永真式。

修改真值赋值 v 中 $A_1, ..., A_n$ 的赋值为 $a_1, ..., a_n$ 得到的赋值记作 $v[A_1/a_1, ..., A_n/a_n]$ 。设 $v' = v[A_1/a_1, ..., A_n/a_n]$ 。 我们有

$$v'(A) = \begin{cases} a_i & if \quad A = A_i, i \in \{1, ..., n\} \\ A^v & if \quad A \notin \{A_1, ..., A_n\} \end{cases}$$

用公式 $\varphi_1, ..., \varphi_n$ 分别替换公式 φ 中的不同命题变元 $A_1, ..., A_n$ 得到的公式记作 $\varphi^{\varphi_1, ..., \varphi_n}_{A_1, ..., A_n}$. 我们有

$$v(\varphi^{\varphi_1, ..., \varphi_n}_{A_1, ..., A_n}) = v[A_1/v(\varphi_1), ..., A_n/v(\varphi_n)](\varphi)$$

设 φ 是由 $\{0, 1, \neg, \wedge, \vee\}$ 生成的公式。将 φ 中的 $\wedge$ 与 $\vee$ 互换、0 与 1 互换等到的公式 φ^*，称为 φ 的对偶式。 对于真值赋值 v 和其相反的真值赋值 v'，我们有 $v(\varphi) = \neg v'(\varphi^*)$ 。 设 φ 与 φ^* 互为对偶式，ψ 与 ψ^* 互为对偶式。如果 $\varphi \Leftrightarrow \psi$，则 $\varphi^* \Leftrightarrow \psi^*$ 。

逻辑公式的合取、析取和否运算满足幂等律、结合律、交换律、分配律、吸收律，德摩根律（对偶关系）。

$A \wedge A$	$=$	A	$A \vee A$	$=$	A
$A \wedge (B \wedge C)$	$=$	$(A \wedge B) \wedge C$	$A \vee (B \vee C)$	$=$	$(A \vee B) \vee C$
$A \wedge B$	$=$	$B \wedge A$	$A \vee B$	$=$	$B \vee A$
$A \wedge (B \vee C)$	$=$	$A \wedge B \vee A \wedge C$	$A \vee (B \wedge C)$	$=$	$A \vee B \wedge A \vee C$
$A \wedge (A \vee B)$	$=$	A	$A \vee (A \wedge B)$	$=$	A
$\neg(A \wedge B)$	$=$	$\neg A \vee \neg B$	$\neg(A \vee B)$	$=$	$\neg A \wedge \neg B$

设 f 是 n 元联结词，$A_1, ..., A_n$ 是不同的命题变元。如果公式 φ 中不出现除 $A_1, ..., A_n$ 之外的的命题变元，且 $\varphi = f(A_1, ..., A_n)$，则称 φ 定义 f。如果存在由 S 生成的公式定义 f，则称 f 可由 S 定义。

设 S 是联结词集合。若每个 n 元 $(n \geq 1)$ 联结词都可由 S 定义，则称 S 为完全集。若 S 的任何真子集都不是完全集，则称 S 为极小完全集。$\{\neg, \wedge, \vee\}$ 是完全集，$\{\neg, \wedge\}$ 是极小完全集。

设 Γ 为公式集合，如果真值赋值 v 满足 Γ 中的每个公式，则称 v 满足 Γ。如果有真值赋值 v 满足 Γ，则称 Γ 是可满足的。否则称 Γ 是不可满足的。

设 Γ 为公式集合，φ 为公式。如果每个满足公式集合 Γ 的真值赋值都满足 φ，则称 φ 是 Γ 的逻辑推论，记作 $\Gamma \models \varphi$。$\Gamma \models \varphi$ 不成立记作 $\Gamma \not\models \varphi$。若 $\Gamma = \{\varphi_1, ..., \varphi_n\}$，则将 $\Gamma \models \varphi$ 写作 $\varphi_1, ..., \varphi_n \models \varphi$。

设 $\varphi_1, ..., \varphi_n, \varphi, \psi$ 为公式。$\models \varphi$ 当且仅当 φ 是永真式。$\varphi_1, ..., \varphi_n \models \varphi$ 当且仅当 $\varphi_1 \wedge \cdots \wedge \varphi_n \rightarrow \varphi$ 是永真式。$\varphi \Leftrightarrow \psi$ 当且仅当 $\varphi \models \psi$ 且 $\psi \models \varphi$。$\Gamma \cup \{\varphi\} \models \psi$ 当且仅当 $\Gamma \models \varphi \rightarrow \psi$。$\Gamma = \{\varphi_1, ..., \varphi_n\}$ 是可满足的当且仅当 $\varphi_1 \wedge \cdots \wedge \varphi_n$ 是可满足的。设 Γ 是公式集合。则 Γ 是不可满足的当且仅当每个公式都是 Γ 的逻辑推论。

§1.2 谓词逻辑

谓词逻辑可以对所考察的命题加以细化，分清主词和谓词，考虑一般和个别情况。谓词逻辑中使用的符号有以下几组：（1）个体变元，简称变元，有无穷多个，用 x, y, z, u, v, w 表示。（2）个体常元，简称常元，用 a, b, c 表示。（3）函数符号，每个符号都有与之相联系的正整数 n，并称该符号为 n 元函数符号，用 f, g, h 表示。（4）谓词符号，每个符号都有与之相联系的正整数 n，并称该符号为 n 元谓词符号，用 A, B, C 表示。（5）量词符号 $\forall$ 和 $\exists$。（6）联结词符号 $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$。（7）左括号（，右括号），点。和逗号，。

设 F 是常元和函数符号的集合。由 F 生成的项定义如下。（1）变元是由 F 生成的项；（2）F 中的常元是由 F 生成的项；（3）若 f 是 F 中的 n 元 $(n \geq 1)$ 函数符号，$t_1, ..., t_n$ 是由 F 生成的项，则 $f(t_1, ..., t_n)$ 是由 F 生成的项。

设 G 是谓词符号的集合。若 $t_1, ..., t_n$ 是由 F 生成的项，A 是 P 中的 n 元谓词符号，则 $A(t_1, ..., t_n)$ 是由 (F, G) 生成的原子公式。

$B = (F, G)$ 上的公式集合，记作 $\mathcal{L}^B$，定义如下。（1）由 (F, G) 生成的原子公式是 $\mathcal{L}^B$ 的公式；（2）若 f 是 $\{\neg, \wedge, \vee, \rightarrow, \leftrightarrow\}$ 中的 n 元 $(n \geq 1)$ 联结词，$\varphi_1, ..., \varphi_n$ 是 $\mathcal{L}^B$ 的公式，则 $f(\varphi_1, ..., \varphi_n)$ 是 $\mathcal{L}^B$ 的公式；（3）若 φ 是 $\mathcal{L}^B$ 的公式，x 是变元，则 $\forall x\varphi$ 和 $\exists x\varphi$ 是 $\mathcal{L}^B$ 的公式。

一个解释 I 由两个部分组成。其一是一个非空集合，称为论域，其二是 B 中符号到论域中的元素、函数、谓词的解释（映射）。设 $I = (D, I_0)$。对于每个常元 a，$I_0(a)$ 为 D 中的一

个元素； 对于每个 n 元函数符号 f， $I_0(f)$ 为 D 中的一个 n 元函数； 对于每个 n 元谓词符号 P， $I_0(P)$ 为 D 中的一个 n 元谓词。

设 I 是一个解释。从所有变元组成的集合到论域 D 的函数称为 I 中的赋值。 修改赋值 σ 中 $x_1,...,x_n$ 的赋值为 $a_1,...,a_n$ 得到的赋值记作 $\sigma[x_1/a_1,...,x_n/a_n]$。 我们有

$$\sigma[x_1/a_1,...,x_n/a_n](x)=\begin{cases} a_i & if \quad x=x_i, i\in\{1,...,n\} \\ \sigma(x) & if \quad x\notin\{x_1,...,x_n\} \end{cases}$$

解释和赋值共同规定了项和公式的意义。 设 σ 是 I 中的赋值。项 t 在解释 I 和赋值 σ 下的意义 $I(t)\sigma$ 定义如下。 （1）若 t 是变元 x，则 $I(t)\sigma=\sigma(x)$； （2）若 t 是常元 a，则 $I(t)\sigma=I_0(a)$； （3）若 t 是 $f(t_1,...,t_n)$，其中 f 是 n 元函数符号， $t_1,...,t_n$ 是项，则 $I(t)\sigma=I_0(f)(I(t_1)\sigma,...,I(t_n)\sigma)$。

设 σ 是 I 中的赋值。公式 φ 在解释 I 和赋值 σ 下的意义 $I(\varphi)\sigma$ 定义如下。 （1）若 φ 是 $P(t_1,...,t_n)$，其中 P 是 n 元谓词符号， $t_1,...,t_n$ 是项，则 $I(\varphi)\sigma=I_0(P)(I(t_1)\sigma,...,I(t_n)\sigma)$；（2）若 φ 是 $\neg\psi$，ψ 是公式，则 $I(\varphi)\sigma=\neg I(\psi)\sigma$；（3）若 φ 是 $\varphi_0\wedge\varphi_1$，则 $I(\varphi)\sigma=I(\varphi_0)\sigma\wedge I(\varphi_1)\sigma$；（4）若 φ 是 $\varphi_0\vee\varphi_1$，则 $I(\varphi)\sigma=I(\varphi_0)\sigma\vee I(\varphi_1)\sigma$；（5）若 φ 是 $\varphi_0\to\varphi_1$，则 $I(\varphi)\sigma=I(\varphi_0)\sigma\to I(\varphi_1)\sigma$；（6）若 φ 是 $\varphi_0\leftrightarrow\varphi_1$， 则 $I(\varphi)\sigma=I(\varphi_0)\sigma\leftrightarrow I(\varphi_1)\sigma$； （7）若 φ 是 $\forall x\psi$， 则 $I(\varphi)\sigma=1$ 当且仅当对于所有 $d\in D$， $I(\psi)\sigma[x/d]=1$； （8）若 φ 是 $\exists x\psi$， 则 $I(\varphi)\sigma=1$ 当且仅当存在 $d\in D$ 使得 $I(\psi)\sigma[x/d]=1$。

如果公式 ψ 在公式 φ 中出现，则称 ψ 为 φ 的子公式。 变元 x 在 $\forall x\varphi$ 或 $\exists x\varphi$ 中的出现为约束出现，并称 $\forall x$ 或 $\exists x$ 的该次出现的辖域为 φ。 如果变元 x 在 φ 中的某次出现是在 φ 的一个子公式中的约束出现，则称 x 的该次出现为在 φ 中的约束出现。 如果变元 x 在 φ 中的某次出现不是约束出现，则称该出现为在 φ 中的自由出现。 在公式 φ 中有自由出现的变元称为 φ 的自由变元，在公式 φ 中有约束出现的变元称为 φ 的约束变元。 φ 中自由变元的集合记为 $Var(\varphi)$。

不出现变元的项称为基项。 没有自由变元的公式称为语句。 没有约束变元的公式称为开公式。 若 $Var(\varphi)=\{x_1,...,x_n\}$，则称公式 $\forall x_1\cdots\forall x_n\varphi$ 为 φ 的闭包。 每个公式的闭包是一个语句，每个语句的闭包是它自己。

若 t 是基项，则对任意 σ,σ' 有 $I(t)\sigma=I(t)\sigma'$，即基项的意义与赋值无关。因此对于基项我们可将 $I(t)\sigma$ 简记为 $I(t)$。 若 φ 是语句，则对任意 σ,σ' 有 $I(\varphi)\sigma=I(\varphi)\sigma'$。因此对于语句我们可将 $I(\varphi)\sigma$ 简记为 $I(\varphi)$。

若 $x_1,...,x_n$ 是不同的变元， $t_1,...,t_n$ 是项，则称 $\{x_1/t_1,...,x_n/t_n\}$ 为代换。 若 t 是项，则 $t\{x_1/t_1,...,x_n/t_n\}$ 是用 $t_1,...,t_n$ 分别替换 t 中 $x_1,...,x_n$ 的所有出现得到的项，记为 $t^{t_1,...,t_n}_{x_1,...,x_n}$。若 φ 是公式，则 $\varphi\{x_1/t_1,...,x_n/t_n\}$ 是用 $t_1,...,t_n$ 分别替换 t 中 $x_1,...,x_n$ 的所有自由出现得到的公式，记为 $\varphi^{t_1,...,t_n}_{x_1,...,x_n}$。 如果在公式 φ 和 $\varphi^{t_1,...,t_n}_{x_1,...,x_n}$ 中变元的约束出现次数相同，则称 $t_1,...,t_n$ 对于 φ 中的 $x_1,...,x_n$ 是可代入的。 若 $t_1,...,t_n$ 对于 φ 中的 $x_1,...,x_n$ 是可代入的，则有

$$I(\varphi^{t_1,...,t_n}_{x_1,...,x_n})\sigma=I(\varphi)\sigma[x_1/I(t_1)\sigma,...,x_n/I(t_n)\sigma]$$

如果解释 I 和 I 中的赋值 σ 使得 $I(\varphi)\sigma=1$，则称解释 I 和赋值 σ 满足 φ，记作 $\sigma\models_I\varphi$。当解释给定时，简记为 $\sigma\models\varphi$。

如果有解释 I 和 I 中的赋值 σ 使得 $\sigma\models_I\varphi$，则称 φ 为可满足式。否则称 φ 为永假式（不可满足式）。 如果 φ 在每个解释中为真，则称 φ 为永真式（逻辑有效式）。

用谓词逻辑公式 $\varphi_1,...,\varphi_i$ 分别替换命题逻辑公式 φ 中的命题变元 $A_1,...,A_n$ 得到的谓词逻辑公式记为 $\varphi^{\varphi_1,...,\varphi_n}_{A_1,...,A_n}$，称为 φ 的替换实例。 命题逻辑永真式的替换实例称为重言式。

设 φ 和 ψ 是公式。如果对于每个解释 I 和 I 中的赋值 σ，$I(\varphi)\sigma = I(\psi)\sigma$，则称 φ 和 ψ 逻辑等价，记为 $\varphi \Leftrightarrow \psi$。 $\varphi \Leftrightarrow \psi$ 当且仅当 $\varphi \leftrightarrow \psi$ 是永真式。 对于 $\forall$ 和 $\exists$，我们有 $\forall x\varphi \Leftrightarrow \neg\exists x\neg\varphi$。

设 Γ 为公式集合，解释 I 和 I 中的赋值 σ 满足 Γ 中的每个公式，则称 I 和 σ 满足 Γ。 如果有解释 I 和 I 中的赋值 σ 满足 Γ，则称 Γ 是可满足的。否则称 Γ 是不可满足的。

设 Γ 为公式集合， φ 为公式。如果每个满足公式集合 Γ 的解释 I 和 I 中的赋值 σ 都满足 φ，则称 φ 是 Γ 的逻辑推论，记作 $\Gamma \models \varphi$。 $\Gamma \models \varphi$ 不成立记作 $\Gamma \not\models \varphi$。 若 $\Gamma = \{\varphi_1, ..., \varphi_n\}$，则将 $\Gamma \models \varphi$ 写作 $\varphi_1, ..., \varphi_n \models \varphi$。

设 $\varphi_1, ..., \varphi_n, \varphi, \psi$ 为公式。$\models \varphi$ 当且仅当 φ 是永真式。$\varphi_1, ..., \varphi_n \models \varphi$ 当且仅当 $\varphi_1 \wedge \cdots \wedge \varphi_n \rightarrow \varphi$ 是永真式。 $\varphi \Leftrightarrow \psi$ 当且仅当 $\varphi \models \psi$ 且 $\psi \models \varphi$。 $\Gamma \cup \{\varphi\} \models \psi$ 当且仅当 $\Gamma \models \varphi \rightarrow \psi$。 $\Gamma = \{\varphi_1, ..., \varphi_n\}$ 是可满足的当且仅当 $\varphi_1 \wedge \cdots \wedge \varphi_n$ 是可满足的。 设 Γ 是公式集合。则 Γ 是不可满足的当且仅当每个公式都是 Γ 的逻辑推论。

§1.3 集合

集合是由一些个体组成的整体。这些个体称为集合的元素。 集合的定义有两种：枚举定义和抽象定义。 枚举定义即是列出所有属于集合的元素。如： $A = \{a, b, c\}$。 抽象定义即是说明属于集合的元素所具有的性质特征。如： $A = \{x \in \mathbf{N} \mid x > 1\}$ 或 $x \in A \Leftrightarrow x > 1$。

两个集合相等，记作 $A = B$，当且仅当他们具有相同的元素。 集合 A 是集合 B 的子集，或说集合 A 包含于集合 B，记作 $A \subseteq B$，当且仅当所有 A 的元素都是 B 的元素。

$$\begin{array}{|lcl|}\hline (A = B) & \Leftrightarrow & \forall x(x \in A \leftrightarrow x \in B) \\ (A \subseteq B) & \Leftrightarrow & \forall x(x \in A \rightarrow x \in B) \\ \hline\end{array}$$

子集关系满足以下性质。

$$\begin{array}{|l|}\hline A \subseteq A \\ A \subseteq B \text{ 且 } B \subseteq C \text{ 则 } A \subseteq C \\ A \subseteq B \text{ 且 } B \subseteq A \text{ 则 } A = B \\ \hline\end{array}$$

不含有任何元素的集合称为空集，记作 $\emptyset$。空集是最小的集合，是唯一的，它包含于任何集合之中。 由有限多个元素构成的集合称为有穷集。由无限多个元素构成的集合称为无穷集。集合 A 的全部子集的集合称为 A 的幂集，记作 2^A，即 $2^A = \{X \mid X \subseteq A\}$。 若 $a \in A$，则 $\{a\} \subseteq A$。若 $A \subseteq B$，则 $A \in 2^B$。 有穷集合 A 的元素个数称为基数，记作 $|A|$。设 A 是有穷集合，则 $|2^A| = 2^{|A|}$。

集合的运算有交、并、差。

$$\begin{array}{|ll|}\hline \text{交} & A \cap B = \{x \mid x \in A \wedge x \in B\} \\ \text{并} & A \cup B = \{x \mid x \in A \vee x \in B\} \\ \text{差} & A - B = \{x \mid x \in A \wedge x \notin B\} \\ \hline\end{array}$$

若 $A \cap B = \emptyset$，则称 A 和 B 是不相交的。 集合 A 和 B 的差集 $A - B$ 又称 B 关于 A 的相对补集。 设 U 是全集或论域，即所有与讨论相关的集合都是该全集的子集。设 A 是 U 的子集， A 关于 U 的相对补集 $U - A$，称为 A 的绝对补集，通常就称补集，记作 $\sim A$。

与逻辑公式的合取、析取和否运算类似，集合的交、并、补运算满足幂等律、结合律、交换律、分配律、吸收律，德摩根律。

任给两个对象 x, y，将它们按规定的顺序构成的序列，称为有序偶，记为 $\langle x, y\rangle$。 有序偶有第一个元与第二个元之分， $\langle x, y\rangle$ 的第一个元是 x，第二个元是 y。有序偶可用集合表示。$\langle x, y\rangle$ 的集合表示为 $\{\{x\}, \{x, y\}\}$。 $\langle x, y\rangle = \langle u, v\rangle$ 当且仅当 $x = u$ 且 $y = v$。

$A \cap A$	$=$	A	$A \cup A$	$=$	A
$A \cap (B \cap C)$	$=$	$(A \cap B) \cap C$	$A \cup (B \cup C)$	$=$	$(A \cup B) \cup C$
$A \cap B$	$=$	$B \cap A$	$A \cup B$	$=$	$B \cup A$
$A \cap (B \cup C)$	$=$	$A \cap B \cup A \cap C$	$A \cup (B \cap C)$	$=$	$A \cup B \cap A \cup C$
$A \cap (A \cup B)$	$=$	A	$A \cup (A \cap B)$	$=$	A
$\sim (A \cap B)$	$=$	$\sim A \cup \sim B$	$\sim (A \cup B)$	$=$	$\sim A \cap \sim B$

有序偶可以推广到 n 重序偶。n 重序偶定义为 $\langle x_1, ..., x_{n-1}, x_n \rangle = \langle \langle x_1, ..., x_{n-1} \rangle, x_n \rangle$ 。$\langle x_1, ..., x_{n-1}, x_n \rangle = \langle y_1, ..., y_{n-1}, y_n \rangle$ 当且仅当 $x_1 = y_1$, . . . , $x_{n-1} = y_{n-1}$ 且 $x_n = y_n$ 。

集合 $A_1, ..., A_n$ 的笛卡尔乘积 $A_1 \times \cdots \times A_n$ 定义为 $A_1 \times \cdots \times A_n = \{\langle a_1, ..., a_n \rangle \mid a_1 \in A_1, ..., a_n \in A_n\}$ 。对于任意有穷集合 $A_1, ..., A_n$ ， $|A_1 \times \cdots \times A_n| = |A_1| \cdot |A_2| \cdots |A_n|$ 。 设 $A = A_1 \times \cdots \times A_n$ 。则第 i 分量函数 $pr_i : A \to A_i$ 定义如下。 $pr_i(\langle a_1, ..., a_n \rangle) = a_i$ 。

§1.4 关系

任何有序偶的集合称为二元关系。 从 X 到 Y 的关系 R 满足 $R \subseteq X \times Y$ 。 若 $\langle x, y \rangle \in R$, 则表示成 xRy , 读作 x 与 y 有关系 R 。若 $\langle x, y \rangle \notin R$, 则表示成 $x\bar{R}y$ 。 一个二元关系可以用一个二元谓词确定。定义 $R = \{\langle x, y \rangle \mid P(x, y)\}$, 即 xRy 当且仅当 $P(x, y)$ 成立。

设 R 是一个关系。R 中所有有序偶的第一个元的集合称为 R 的定义域, 记作 $dom(R)$ 。R 中所有有序偶的第二个元的集合称为 R 的值域, 记作 $ran(R)$ 。 集合 X 到 X 的关系称为 X 上的二元关系。关系的性质由关系中包含的所有有序偶所确定。 记 $\forall x_1 \cdots \forall x_n (x_1 \in X \wedge \cdots \wedge x_n \in X \to \varphi)$ 为 $\forall x_1, ..., x_n \in X.\varphi$ 。设 R 是非空集合 X 上的关系。

自反性	$\forall x \in X.(xRx)$
反自反性	$\forall x \in X.(x\bar{R}x)$
对称性	$\forall x, y \in X.(xRy \to yRx)$
反对称性	$\forall x, y \in X.(xRy \wedge yRx \to x = y)$
传递性	$\forall x, y, z \in X.(xRy \wedge yRz \to xRz)$

如果 R 和 S 是 X 到 Y 的二元关系, 则 $R \cap S$, $R \cup S$, $R - S$, $\sim S$ 都是 X 到 Y 的二元关系, 且

$x(R \cap S)y$	$\Leftrightarrow$	$xRy \wedge xSy$
$x(R \cup S)y$	$\Leftrightarrow$	$xRy \vee xSy$
$x(R - S)y$	$\Leftrightarrow$	$xRy \wedge x\bar{S}y$
$x(\sim S)y$	$\Leftrightarrow$	$x\bar{S}y$

设 R 是 X 到 Y 的关系。R 的逆关系是 Y 到 X 的关系, 记作 R^{-1} , 定义为 $R^{-1} = \{\langle y, x \rangle \in Y \times X \mid \langle x, y \rangle \in R\}$ 。

设 R 是 X 到 Y 的关系， S 是 Y 到 Z 的关系。 $R \circ S$ 是 X 到 Z 的关系, 称为 R 和 S 的复合关系, 定义为 $R \circ S = \{\langle x, z \rangle \in X \times Z \mid \exists y \in Y.(xRy \wedge ySz)\}$ 。 $\circ$ 称为关系的复合运算。在不引起混淆的情况下, 复合关系的运算符常省略不写。关系的复合运算满足结合律。 设 R 是 X 到 Y 的关系， S 是 Y 到 Z 的关系。则有 $(R \circ S)^{-1} = S^{-1} \circ R^{-1}$ 。

设 R 是 X 上的二元关系， $n \in \mathbf{N}$ 。 R 的 n 次幂, 记作 R^n , 定义如下。 $R^0 = I_X = \{\langle x, x \rangle \mid x \in X\}$ 是集合 X 上的恒等关系； $R^{n+1} = R^n \circ R$ 。

设 R 是 X 上的二元关系，关系 R' 是 R 的自反（对称、传递）闭包当且仅当 （1） R' 是自反（对称、传递）的； （2） $R \subseteq R'$； （3）对于任何自反（对称、传递）关系 R''，如果 $R \subseteq R''$，则 $R' \subseteq R''$。 用 $r(R), s(R), t(R)$ 分别表示 R 的自反闭包、对称闭包、传递闭包。我们有

$$\begin{array}{lcl} r(R) & = & R \cup I_X \\ s(R) & = & R \cup R^{-1} \\ t(R) & = & \bigcup_{i=1}^{\infty} R^i \end{array}$$

为书写方便，关系 R 的传递闭包通常记为 R^+。 R 的自反传递闭包通常记为 R^*。

满足自反性、对称性和传递性的一个非空集合上的关系称为等价关系。 设 R 是集合 X 上的等价关系。对于任意 $x \in X$，集合 $[x]_R$ 定义如下：$[x]_R = \{y \in X \mid yRx\}$。 称 $[x]_R$ 为由 x 所代表的等价类。用 X/R 表示 R 等价类的集合 $\{[x]_R \mid x \in X\}$，称 X/R 为 X 模 R 的商集。

设 A 是非空集合 S 子集的聚合。对于每个集合 $B \in A$， $B \neq \emptyset$ 且 $\bigcup A = S$。则称集合 A 是 S 的一个覆盖。 若 A 是 S 的一个覆盖，且对任意 $B, C \in A$， $B \neq C$ 则 $B \cap C = \emptyset$，则称 A 是 S 的一个划分。

任何一个 X 上的等价关系 R 都定义了 X 的一个划分，即 X/R。 任何 X 的一个划分 $A = \{A_1, ..., A_n\}$ 都定义了一个等价关系 R，即 xRy 当且仅当 x, y 同在一个 A_i 中。

满足自反性、反对称性和传递性的一个非空集合上的关系称为偏序关系。 如果 $\leq$ 是 X 上的偏序关系，那么有序偶 $\langle P, \leq \rangle$ 表示偏序集合。

设 $\langle P, \leq \rangle$ 是偏序集合。如果对于每一个 $x, y \in P$ 都有 $x \leq y$ 或 $y \leq x$，则称 $\leq$ 上为 P 上的全序或线序，称 $\langle P, \leq \rangle$ 为全序集合或链。

设 $\langle P, \leq \rangle$ 是偏序集合，并且 $A \subseteq P$。设 $a \in A, b \in P$。

a 为 A 的最大元：	$\forall x \in A.(x \leq a)$
a 为 A 的最小元：	$\forall x \in A.(a \leq x)$
a 为 A 的极大元：	$\forall x \in A.(a \neq x \rightarrow a \not\leq x)$
a 为 A 的极小元：	$\forall x \in A.(a \neq x \rightarrow x \not\leq a)$
b 为 A 的上界：	$\forall x \in A.(x \leq b)$
b 为 A 的下界：	$\forall x \in A.(b \leq x)$
b 为 A 的最小上界：	b 是 A 的上界，且对于每一个 A 的上界 b'，有 $b \leq b'$
b 为 A 的最大下界：	b 是 A 的下界，且对于每一个 A 的下界 b'，有 $b' \leq b$

一个偏序集合的最大元，最小元，最小上界，最大下界，如果存在，则是唯一的。 最小上界，也称上确界，记作 $\sqcup$， lub 或 sup，最大下界，也称下确界，记作 $\sqcap$， glb 或 inf。

一个偏序集合 $\langle P, \leq \rangle$，如果它的每一个非空子集都有一个最小元，则称 $\leq$ 为良序的，称 $\langle P, \leq \rangle$ 为良序集合。 每一个良序集合都是全序集合，但全序集合未必都是良序集合，而每一个有限的全序集合都是良序集合。

一个偏序集合 $\langle P, \leq \rangle$，如果它的每一个非空子集都有一个极小元，则称 $\leq$ 为良基的 (Well Founded)，称 $\langle P, \leq \rangle$ 为良基集合。 一个集合是良基集合当且仅当该集合中不存在无限递减的序列。

设 $\langle P, \leq \rangle$ 为良基集合。记 $x \leq y$ 且 $x \neq y$ 为 $x < y$。以下为良基归纳推理 (Noetherian Induction)。

$$\text{若 } \forall x' \in P.(\forall x \in P.(x < x' \rightarrow \varphi(x)) \rightarrow \varphi(x')) \text{ 则 } \forall x \in P.\varphi(x) \text{。}$$

设 $\langle P_1,\leq_1\rangle,...,\langle P_n,\leq_n\rangle$ 为偏序，$P=P_1\times\cdots\times P_n$。则偏序 $\langle P_1,\leq_1\rangle,...,\langle P_n,\leq_n\rangle$ 的笛卡尔积 $\langle P,\leq\rangle$ 是一个偏序，其中 $\leq$ 定义如下。对所有 $a,b\in P$，$a\leq b$ 当且仅当对所有 $i\in\{1,...,n\}$，有 $pr_i(a)\leq_i pr_i(b)$。设 $S\subseteq P_1\times\cdots\times P_n$。则 $\sqcup S$ 存在当且仅当对于所有 $i\in\{1,...,n\}$，$\sqcup pr_i(S)$ 存在，且 $\sqcup S=\langle\sqcup pr_1(S),...,\sqcup pr_n(S)\rangle$。

§1.5 函数

设 f 是集合 X 到集合 Y 的关系。如果对每一个 $x\in X$ 存在唯一的 $y\in Y$ 使得 $\langle x,y\rangle\in f$，则称 f 为 X 到 Y 的一个函数。记为 $f:X\to Y$。X 称为 f 的定义域，Y 称为 f 的值域。

对于函数 $f:X\to Y$，如果 $\langle x,y\rangle\in f$，则称 x 为自变量，y 是函数 f 在 x 处的值，也称 y 为在 f 作用下 x 的象，而称 x 为 y 的一个象源。通常用 $y=f(x)$ 表示 $\langle x,y\rangle\in f$。

设函数 $f:X\to Y$ 且 $A\subseteq X$，则 $f\cap(A\times Y)$ 是从 A 到 Y 的函数，称为 f 受限制于 A，记为 $f|A$。集合 $\{f(x)\mid x\in A\}$ 称为 A 在 f 下的象，记为 $f(A)$。

若 $X'\subseteq X$，且 $f:X'\to Y$ 是 X' 到 Y 的函数，则称 f 为 X 到 Y 的偏函数。为区别于偏函数，函数又称全函数。

设 $f:X\to Y$ 和 $g:Y\to Z$ 是两个函数，则 f 和 g 的复合函数 $g\circ f$ 是一个从 X 到 Z 的函数，且对于所有的 $x\in X$，$(g\circ f)(x)=g(f(x))$。函数的复合满足结合律。

若 $f:X\to X$ 是一个函数，则 f 能够对自身复合任意多次。f 对自身复合任意多次的定义如下。$f^0(x)=I_X(x)$；$f^{n+1}(x)=f(f^n(x))$。

记 $\exists x_1\cdots\exists x_n(x_1\in X\wedge\cdots\wedge x_n\in X\wedge\varphi)$ 为 $\exists x_1,...,x_n\in X.\varphi$。设 $f:X\to Y$ 是一个函数。

f 是满射的	$\forall y\in Y.\exists x\in X.(f(x)=y)$
f 是入射的	$\forall x_1,x_2\in X.(f(x_1)=f(x_2)\to x_1=x_2)$
f 是双射的	f 是满射的且是入射的

若 $f:X\to Y$,$g:Y\to Z$ 都是满射函数，则 $g\circ f$ 也是满射函数；若 $f:X\to Y$,$g:Y\to Z$ 都是入射函数，则 $g\circ f$ 也是入射函数；若 $f:X\to Y$,$g:Y\to Z$ 都是双射函数，则 $g\circ f$ 也是双射函数。

设 f 是双射的。它的反函数是 f 的逆关系，记作 f^{-1}。若 $f:X\to Y$ 是双射的，则其反函数 $f^{-1}:Y\to X$ 也是双射的。若 $f:X\to Y$,$g:Y\to Z$ 都是双射函数，则 $(g\circ f)^{-1}$ 也是双射函数，且 $(g\circ f)^{-1}=f^{-1}\circ g^{-1}$。

设 U 是全集，$A\subseteq U$。A 的特征函数 $\Psi_A:U\to\{0,1\}$ 定义如下。

$$\Psi_A(x)=\begin{cases}1 & if \quad x\in A\\ 0 & if \quad x\notin A\end{cases}$$

设 U 是全集，$A,B\subseteq U$。则对所有 $x\in U$，以下等式成立。

$\Psi_{A\cap B}(x)$	$=$	$\Psi_A(x)\cdot\Psi_B(x)$
$\Psi_{A\cup B}(x)$	$=$	$\Psi_A(x)+\Psi_B(x)-\Psi_{A\cap B}(x)$
$\Psi_{\sim A}(x)$	$=$	$1-\Psi_A(x)$

X 到 Y 的所有全函数的集合记作 $(X\to Y)$。设 X 是一个集合，$\langle Y,\leq\rangle$ 是一个偏序，$f,g:X\to Y$ 是两个函数。$f\leq g$ 当且仅当对于所有 $x\in X$，$f(x)\leq g(x)$。$\langle(X\to Y),\leq\rangle$ 构成一个偏序。

设 $S\subseteq(X\to Y)$ 是一个 X 到 Y 的函数的集合。设 $x\in X$。Y 的子集 $\{f(x)\mid f\in S\}$ 记作 $S(x)$。$\sqcup S$ 存在当且仅当对于所有 $x\in X$，$\sqcup S(x)$ 存在，且对于所有 $x\in X$，$(\sqcup S)(x)=\sqcup S(x)$。

设 X 是一个集合。记 X 的大小为 n 的子集的集合为 $[X]^n$ 。一个 $[X]^n$ 的 k 染色函数 C 是一个 $[X]^n$ 到一个大小为 k 的集合的函数。称 H 为 C 的齐性集合当且仅当 C 作用在任意 $[H]^n$ 中的元素所得的值是一样的。以下结论称为拉姆齐定理 (Infinite Ramsey Theorem) 。

设 N 是自然数集合。任意 $[N]^n$ 的 k 染色函数都有一个无穷的齐性集合。

§1.6 完全偏序和格上的不动点

一个偏序 $\langle X, \leq \rangle$ ，如果它 X 有最小元，且对于 X 上的每一条链 $S \subseteq X$ ， $\sqcup S$ 都存在，则称 $\langle X, \leq \rangle$ 为全偏序。 X 的最小元通常记作 $\perp_X$ 或 $\perp$ 。

任何有最小元且只包含有穷链的偏序是完全偏序。如果 $\langle P_1, \leq_1 \rangle, ..., \langle P_n, \leq_n \rangle$ 是完全偏序，则其笛卡尔积 $\langle P_1 \times \cdots \times P_n, \leq \rangle$ 是完全偏序。 如果 X 是一个集合， $\langle Y, \leq \rangle$ 是一个完全偏序，则 $\langle (X \to Y), \leq \rangle$ 是完全偏序。

设 $\langle X, \leq \rangle$ 是一个完全偏序， $Y \subseteq X$ 。 $\langle Y, \leq \rangle$ 是 $\langle X, \leq \rangle$ 的子完全偏序，当且仅当 $\langle Y, \leq \rangle$ 是一个完全偏序，且对于 Y 上的每一条链 S ，都有 $\sqcup_Y S = \sqcup_X S$ 。

设 $\langle X, \leq_1 \rangle$ 和 $\langle Y, \leq_2 \rangle$ 是偏序， $f : X \to Y$ 是函数。 f 是单调的当且仅当 $\forall x_1, x_2 \in X.(x_1 \leq_1 x_2 \to f(x_1) \leq_2 f(x_2))$ 。

设 $\langle X, \leq_1 \rangle$ 是偏序， $\langle Y, \leq_2 \rangle$ 是完全偏序。从 X 到 Y 的单调函数的集合构成 $\langle (X \to Y), \leq_2 \rangle$ 的一个子完全偏序。

设 $\langle X, \leq_1 \rangle$ 和 $\langle Y, \leq_2 \rangle$ 是完全偏序， $f : X \to Y$ 是函数。 f 是连续的 (Scott-Continous) 当且仅当对 X 的每一条链 $S \subseteq X$ ， $\sqcup f(S)$ 都存在，且 $\sqcup f(S) = f(\sqcup S)$ 。 连续函数的集合记作 $[X \to Y]$ 。

设 $\langle X, \leq_1 \rangle$ 和 $\langle Y, \leq_2 \rangle$ 是完全偏序。从 X 到 Y 的连续函数的集合 $[X \to Y]$ 构成 $\langle (X \to Y), \leq \rangle$ 的一个子完全偏序。

设 $\langle X, \leq_1 \rangle$ 和 $\langle Y, \leq_2 \rangle$ 是完全偏序， $f : X \to Y$ 是函数。 f 是连续的当且仅当 f 是单调的且对 X 的每一条链 $S \subseteq X$ ， $f(\sqcup S) \leq \sqcup f(S)$ 。

设 $\langle X, \leq \rangle$ 是偏序， $f : X \to X$ 是函数。若 $f(x) = x$ ，则称 x 是 f 的不动点。 若 x 是 f 的不动点，且对任意 f 的不动点 y ，都有 $x \leq y$ ，则称 x 是 f 的最小不动点。 f 的最小不动点记作 μf 。 若 x 是 f 的不动点，且对任意 f 的不动点 y ，都有 $y \leq x$ ，则称 x 是 f 的最大不动点。 f 的最大不动点记作 νf 。

为了能够清楚地说明 f 的受不动点标记约束的自变量是 x ， μf 有时写成 $\mu x.f$ ， νf 写成 $\nu x.f$ 。

设 $\langle X, \leq \rangle$ 是完全偏序， $f : X \to X$ 是连续函数。则 f 有最小不动点，且可表示如下 (Kleene Fixpoint Theorem) 。

$$\mu f = \sqcup \{ f^i(\perp) \mid i \in \mathbf{N} \}$$

设 $\langle X, \leq \rangle$ 是完全偏序， $f : X \to X$ 是连续函数， $x \in X$ 。若 $f(x) \leq x$ ，则 $\mu f \leq x$ 。

设 $\langle X, \leq \rangle$ 是完全偏序， $\varphi : X \to \{0, 1\}$ 是一个谓词。 φ 是相容的当且仅当对 X 的每一条链 S ，若 $\bigwedge_{x \in S} \varphi(x)$ 为真，则 $\varphi(\sqcup S)$ 为真。

设 $\langle X, \leq \rangle$ 是完全偏序， $\varphi : X \to \{0, 1\}$ 是一个相容谓词。以下为不动点归纳推理 (Scott Induction) 。

$$\text{若 } \varphi(\perp) \text{ 且 } \forall x \in X.(\varphi(x) \to \varphi(f(x))) \text{ ，则 } \varphi(\mu f) \text{ 。}$$

一个偏序 $\langle X, \leq \rangle$ ，如果任意两个 X 中的元素都有最小上界和最大下界，则称 $\langle X, \leq \rangle$ 为格。

一个偏序 $\langle X, \leq \rangle$ ，如果任 X 中的任意子集都有最小上界，则称 $\langle X, \leq \rangle$ 为完全格。一个偏序 $\langle X, \leq \rangle$ 是完全格当且仅当其任意子集都有最大下界。

设 $\langle X, \leq \rangle$ 是完全格， $f : X \to X$ 是单调函数。则 f 有非空的不动点集，且该集构成一个完全格， f 有最小和最大不动点，且可表示如下 (Knaster-Tarski Fixpoint Theorem) 。

$$\mu f = \sqcap \{x \in X \mid f(x) \leq x\}$$
$$\nu f = \sqcup \{x \in X \mid x \leq f(x)\}$$

§1.7 有向图

有向图 D 是一有序偶 $\langle V, E \rangle$ ，其中 V 是一非空集合， E 是 V 上的一个二元关系。分别称 V 和 E 为有向图 D 的顶点的集合和边的集合。为表示的直观性，边的集合 E 有时写成 $\to$ 。关系 $\to$ 的传递闭包和传递自反闭包分别记做 $\stackrel{+}{\to}$ 和 $\stackrel{*}{\to}$ 。

设有两个图 $D = \langle V, E \rangle$ 和 $D' = \langle V', E' \rangle$ 。若 $V \subseteq V'$ 且 $E \subset E'$ ，则称 D 为 D' 的子图，并表示为 $D \subseteq D'$ 。 若 $V \subseteq V'$ 且 $E = \{\langle u, v \rangle \in E' \mid u, v \in V\}$ ，则称 D 为 D' 的导出子图。

在有向图 $D = \langle V, E \rangle$ 中，把边的一个序列 $(e_1, ..., e_n)$ 称为一条通路，其中 $e_i = \langle v_i, v_{i+1} \rangle \in E$ （ $i = 1, ..., n$ ）。 通路 $(e_1, ..., e_n)$ 的长度为出现在通路中的边的次数，记作 $|(e_1, ..., e_n)|$ 。通路 $(e_1, ..., e_n)$ 通常也用顶点序列 $(v_1, ..., v_{n+1})$ 表示。

对于有向图的一条通路，如果每条边的出现不超过一次，则称该通路为简单通路。如果每个顶点的出现不超过一次，则称该通路为基本通路。基本通路一定是简单通路。通过有向图中所有顶点的通路称为完备通路。

如果一条通路的开始和终结于同一顶点，则称该通路为回路。如果该回路中每条边的出现不超过一次，则称该回路为简单回路。如果该回路除去最后一个点的通路中每个顶点的出现不超过一次，则称该回路为基本回路。通过有向图中所有顶点的回路称为完备回路。

如果存在从顶点 u 到顶点 v 的通路，则称顶点 u 可以到达顶点 v ，即 u, v 满足 $u \stackrel{+}{\to} v$ 。如果存在从顶点 u 到顶点 v 的通路，则存在从顶点 u 到顶点 v 的基本通路。 在一个有 n 个顶点的有向图中，任何基本通路的长度都不超过 $n - 1$ ，任何基本回路的长度都不超过 n 。

一个有向图 $D = \langle V, E \rangle$ ，如果对它的每一对顶点 u 和 v ，可以从 u 到达 v 并且可以从 v 到达 u ，则称 D 为强连通图。有向图 D 是强连通的当且仅当 D 有完备回路。称强连通图为非平凡的，当且仅当该图至少有两个顶点或只有一个顶点且有自环。

在有向图 D 中， $A \subseteq D$ 是 D 的一个极大强连通导出子图，当且仅当 A 是 D 的导出子图， A 是强连通的，且对于任意的 D 的强连通的子图 $B \subseteq D$ ，若 $A \neq B$ 则 $A \not\subseteq B$ 。 在有向图 D 中， D 的一个极大强连通导出子图，称为 D 的一个强连通分图。

设 $D = \langle V, E \rangle$ 为有向图。设 $A \subseteq V$ 。从集合 A 可达的顶点的集合（包括 A ）为 $\{b \mid a \stackrel{*}{\to} b, a \in A\}$ ，记为 $rh(A)$ 。可达 A 的顶点的集合为 $\{b \mid b \stackrel{*}{\to} a, a \in A\}$ ，记为 $rh^b(A)$ 。 若 D 是强连通图，则 $rh(A) = rh^b(A) = V$ 。

§1.8 练习

1. 设 $\langle X, \leq_1 \rangle$ 和 $\langle Y, \leq_2 \rangle$ 是完全偏序。 $f : X \to Y$ 是函数。证明：如果 X 只包含有穷链，则 f 是连续的当且仅当 f 是单调的。

2. 设 $D = \langle V, E \rangle$ 为有向图且 V 是有穷集合。设 $A \subseteq V$ 。定义 $E(Z) = \{b \mid E(a, b), a \in Z\}$ 。将 $A \cup E(Z)$ 和 $A \cup E^{-1}(Z)$ 看成是自变量为 Z 的 $(2^V \to 2^V)$ 中的函数，证明： $rh(A) = \bigcup_{i=0}^{\infty} E^i(A) = \mu Z.(A \cup E(Z))$ 且 $rh^b(A) = \bigcup_{i=0}^{\infty} (E^{-1})^i(A) = \mu Z.(A \cup E^{-1}(Z))$ 。

§2 状态迁移模型

本章介绍以抽象状态的迁移为特点的模型。 模型的基本要素为抽象状态和状态迁移关系以及初始状态。 本章的算法仅限于有穷状态系统，而推理方法则不限于有穷状态系统。

基本符号: 设 S 是个集合。

S 的幂集记作 2^S 。

无穷序列 $s_0s_1s_2.....$ 记作 $[s_i]_{i\geq 0}$ 。

有穷序列 $s_0s_1s_2...s_n$ 记作 $[s_i]_{i=0}^n$ 。

S^ω 表示 S 上的无穷序列的集合，即 $S^\omega = \{[s_i]_{i\geq 0} \mid \forall i \geq 0, s_i \in S\}$ 。

依上下文关系， S^n 可表示 S 上的 n 元组的集合 $\{(s_1, ..., s_n) \mid s_1, ..., s_n \in S\}$ 或长度为 n 的有穷序列的集合 $\{[s_i]_{i=0}^{n-1} \mid s_0, ..., s_{n-1} \in S\}$ 。

若 $\rightarrow \subseteq S \times S$ 为 S 上的二元关系，则 $\xrightarrow{*}$ 表示 $\rightarrow$ 的传递自反闭包。

集合的运算包括交、并、差，记作 $\cap, \cup, \setminus$ ，为方便起见，有时写作 $*, +, -$ 。

设 S 为全集。集合 A 的补为全集与 A 的差 $S \setminus A$ ，亦记作 $\neg A$ 。

若 $Y \subseteq S \times S$ 且 $X \subseteq S$ ，用 $Y|X$ 表示 $Y \cap (X \times X)$ 。

若 $G = (V, E)$ 是有向图，依上下文关系，为方便起见， V 的子集 B 亦可表示顶点集合为 B 的导出子图 $(B, E|B)$ 。

用 $e_1, ..., e_k$ 分别替换 φ 中的 $x_1, ..., x_k$ 记作 $\varphi(e_1/x_1, ..., e_k/x_k)$ ，亦写作 $\varphi_{x_1,...,x_k}^{e_1,...,e_k}$ 。

§2.1 Kripke 模型

Definition 2.1 *一个 Kripke 模型是一个三元组 $\langle S, R, I\rangle$ 其中 S 为状态集合， $R \subseteq S \times S$ 为 S 上的完全迁移关系， $I \subseteq S$ 为初始状态集。*

$R \subseteq S \times S$ 是完全的，即 $\forall s.\exists s'.(s, s') \in R$ 。

我们也称 Kripke 模型为系统。

用 $s{\rightarrow}s'$ 表示存在从 s 到 s' 的迁移，即 $(s, s') \in R$ 。

假定 $K = \langle S, R, I\rangle$ 为给定模型。

后继状态: 若 $s{\rightarrow}s'$ ，则称 s' 为 s 的后继状态。状态 s 的后继状态的集合为 $\{s' \mid s{\rightarrow}s'\}$ ，记作 $R(s)$ 。状态集合 A 的后继状态集合为 $\bigcup_{s\in A} R(s)$ ，记作 $R(A)$ 。

可达状态: 若 $s\xrightarrow{*}s'$ ，则称 s' 为 s 的可达状态。 状态 s 的可达状态的集合为 $\{s' \mid s\xrightarrow{*}s'\}$ ，记作 $rh(s)$ 。状态集合 A 的可达状态集合为 $\bigcup_{s\in A} rh(s)$ ，记作 $rh(A)$ 。若 B 中有 s 的可达状态，则称 B 由 s 可达。若 B 中有 A 的可达状态，则称 B 由 A 可达。 K 的可达状态集合为 $rh(I)$ 。

路径: K 的有穷路径是 S 上的满足对任意 $0 \leq i \leq n-1$ 都有 $s_i{\rightarrow}s_{i+1}$ 的有穷序列 $[s_i]_{i=0}^n$ 。 K 的无穷路径是 S 上的满足对任意 $i \geq 0$ 都有 $s_i{\rightarrow}s_{i+1}$ 的无穷序列 $[s_i]_{i\geq 0}$ 。

计算与行为: K 的计算序列 (简称为计算) 是 K 上的满足 $s_0 \in I$ 的无穷路径 $[s_i]_{i\geq 0}$ 。 K 的计算的集合称为 K 的行为，记作 $[[K]]$ 。

状态的满足关系: 称状态 s 满足 A，当且仅当 $s \in A$。因而集合 A 中的状态都满足 A，且满足 A 的状态都在 A 中。称集合 B 满足 A，当且仅当 B 中的所有状态都满足 A，即 $B \subseteq A$。为方便叙述，满足 A 的状态称为 A 状态。若一条路径 (或一个计算) 上的所有状态都满足 A，则称该路径 (计算) 为 A 路径 (计算)。若一条路径 (或一个计算) 上有满足 A 的状态，则称该路径 (计算) 能到达 A 状态。

§2.1.1 安全性质与可达性质的分析

Definition 2.2 (安全性质) 状态集合 $A \subseteq S$ 称为 K 的安全性质，当且仅当 K 的所有计算都是 A 计算。

Proposition 2.1 状态集合 $A \subseteq S$ 是 K 的安全性质当且仅当 $rh(I) \subseteq A$。

Definition 2.3 (可达性质) 状态集合 $A \subseteq S$ 称为 K 的可达性质，当且仅当 $[[K]]$ 中有能到达 A 状态的计算。

Proposition 2.2 状态集合 $A \subseteq S$ 是 K 的可达性质当且仅当 $rh(I) \cap A \neq \emptyset$。

Corollary 2.1 状态集合 $A \subseteq S$ 是 K 的安全性质，当且仅当状态集合 $\neg A$ 不是 K 的可达性质。

可达性分析: 给定 Kripke 模型 $K = \langle S, R, I\rangle$ 和集合 $A \subseteq S$。A 是否是 K 的可达性质可通过对模型所有可达状态进行遍历来验证。这样的算法称为可达性分析算法。

集合类型: 对于集合类型的变量，我们记空集为 $\{\}$。以下操作运算分别代表在集合 w 中添加元素 s，从集合中选取元素，从集合中移除元素 s。

$w.put(s)$
$w.get()$
$w.remove(s)$

可达性分析算法: 给定 Kripke 模型 $K = \langle S, R, I\rangle$ 和集合 $A \subseteq S$。以下算法可分析 A 的可达性。

```
bool ReachabilityAnalysis(K,A)
{   w:=I;
    repeat until w={};
        s:=w.get(); if (s in A) return true;
        visited[s]:=true;
        for each (s' in R(s)), if (visited[s']=false) w.put(s');
        w.remove(s);
    return false;
}
```

Proposition 2.3 状态集合 $A \subseteq S$ 是 K 的可达性质当且仅当 $ReachabilityAnalysis(K, A)$ 为 $true$。

安全性质: 安全性质与可达性质是对偶性质。根据推论 2.1 和命题 2.3，我们有以下结论。

Proposition 2.4 状态集合 $A \subseteq S$ 是 K 的安全性质当且仅当 $ReachabilityAnalysis(K, \neg A)$ 为 $false$。

安全性质的推理: 若 $R(A) \subseteq A$，则称 A 为迁移不变量。关于安全性质，我们有如下推理规则。设 A', A 为状态集合。

$$\frac{\begin{array}{l} I \text{ 满足 } A' \\ A' \text{ 是迁移不变量} \\ A' \text{ 满足 } A \end{array}}{A \text{ 是安全性质}}$$

可靠性与完备性: 以上推理规则是可靠的且完备的。假定前提成立，由前两条我们知道 $rh(I) \subseteq A'$，又由第三条我们知道 $rh(I) \subseteq A$，所以 A 是安全性质，因而推理规则是可靠的。假定 A 是安全的，取 $A' = rh(I)$，我们就可保证前提条件的成立，由此可证 A 是安全性质，因而推理规则是完备的。

§2.1.2 必达性质与可免性质的分析

Definition 2.4 (必达性质) 状态集合 $A \subseteq S$ 称为 K 的必达性质，当且仅当 K 的所有计算都能到达 A 状态。

Proposition 2.5 状态集合 $A \subseteq S$ 是 K 的必达性质当且仅当 $\forall B \subseteq S.((B \cap I \neq \emptyset) \wedge (\forall s \in B.(\exists s' \in B.(s \rightarrow s'))) \rightarrow (B \cap A \neq \emptyset))$。

Definition 2.5 (可免性质) 状态集合 $A \subseteq S$ 称为 K 的可免性质，当且仅当 $[[K]]$ 中有 K 的 $\neg A$ 计算。

Proposition 2.6 状态集合 $A \subseteq S$ 是 K 的可免性质当且仅当 $\exists B \subseteq S.((B \cap I \neq \emptyset) \wedge (\forall s \in B.(\exists s' \in B.(s \rightarrow s'))) \wedge (B \cap A = \emptyset))$。

Corollary 2.2 状态集合 $A \subseteq S$ 是 K 的必达性质，当且仅当状态集合 A 不是 K 的可免性质。

可免性分析: 给定 Kripke 模型 $K = \langle S, R, I \rangle$ 和集合 $A \subseteq S$。A 是否是 K 的可免性质可通过对模型的强连通图的分析来验证，即考察模型是否可由 $\neg A$ 有穷路径到达由 $\neg A$ 状态组成的非平凡强连通图。

栈类型: 对于栈类型的变量，我们记空栈为 $[\,]$。以下操作运算分别代表在栈 w 中添加元素 s，从栈中取出 (最后添加到栈中的) 元素。

$w.push(s)$
$w.pop()$

强连通分量: 给定一个有向图 $G = (V, E)$。以下算法 (称为 Tarjan 算法) 可计算 G 的强连通分量。

```
scclist scctarjan(G)
{   gscclist:={}; gindex:=0; gstack:=[];
    for (each v in V) if (v->index=undefined) strongconnect(v);
    return gscclist;
}
void strongconnect(v)
{   v->index:=v->lowlink:=gindex;  gindex:=gindex+1; gstack.push(v->id);
```

```
    for (each (v,w) in E)
        if (w->index=undefined)
            strongconnect(w);
            v.lowlink:=min(v.lowlink,w.lowlink);
        else if (w in gstack)
            v.lowlink:=min(v.lowlink, w.index);
    if (v.lowlink=v.index)
        currentscc:={};
        repeat w:=gstack.pop(); currentscc.putElement(w); until (w=v);
        gscclist.putElement(currentscc);
}
```

可免性分析算法: 假定 $nontrivial(G, e)$ 为检查 G 的强连通分量 e 是否非平凡的函数。 给定 $K = \langle S, R, I\rangle$ 和集合 $A \subseteq S$ 。以下算法分析可分析可免性。

```
bool AvoidabilityAnalysis(K,A)
{   S':=S-A; R'=R|S'; I':=I-A; G:=(S',R'); K':=(S',R',I');
    scclist:=scctarjan(G);
    w:={}; for (each e in scclist) if (nontrivial(G,e)) w:=(w+e);
    return ReachbilityAnalysis(K',w);
}
```

Proposition 2.7 状态集合 $A \subseteq S$ 是 K 的可免性质当且仅当 $AvoidabilityAnalysis(K, A)$ 为 $true$ 。

必达性质: 必达性质与可免性质是相反的关系。根据推论 2.2 和命题 2.7 ，我们有以下结论。

Proposition 2.8 状态集合 $A \subseteq S$ 是 K 的必达性质当且仅当 $AvoidabilityAnalysis(K, A)$ 为 $false$ 。

必达性质的推理: 关于必达性质，我们有如下推理规则。设 $X_0, X_1, ..., X_n, A$ 为状态集合。

$$\frac{\begin{array}{l} I \subseteq X_0 \\ \forall i \in \{0, ..., n-1\}.(R(X_i \setminus A) \rightarrow X_{i+1}) \\ X_n \subseteq A \end{array}}{A \text{ 是必达性质}}$$

可靠性与完备性: 以上推理规则是可靠的且对于有穷状态系统是完备的。假定前提成立且结论不成立，那么由结论不成立，我们有一条由 I 状态出发的无穷 $\neg A$ 路径，这条路径必然有状态出现在任一 X_i 中，与 X_n 满足 A 矛盾，所以前提成立则结论成立，因而推理规则是可靠的。假定 A 是必达性质，由于只考虑有穷状态系统，我们可以假定状态个数为 n ，那么取 $X_0 = I$ 和 $X_{i+1} = R(X_i \setminus A)$ ，则 X_n 为空 (否则存在一条无限循环的 $\neg A$ 路径) ，由此可证 A 是必达性质，因而推理规则是完备的。

§2.2 公平 Kripke 模型

对于 Kripke 模型而言，一个计算就是以一个初始状态为起点的满足状态迁移关系的无穷状态序列。为了更精确地描述模型的计算，我们对计算做一定限制以排除部分满足状态转换关系的无穷状态序列。 为达到这个效果，我们在结构中加入新的成分，设置公平计算的条件。

Definition 2.6 *一个公平 Kripke 模型是一个四元组 $\langle S, R, I, F\rangle$ 其中 $\langle S, R, I\rangle$ 是一个 Kripke 模型， $F \subseteq 2^S$ 为公平性约束的有穷集合。*

假定 $K = \langle S, R, I, F\rangle$ 为给定模型。

公平路径: 公平 Kripke 模型 K 上的一条路径即 Kripke 模型 $\langle S, R, I\rangle$ 上的一条路径。定义 $inf(\pi)$ 为无限多次出现在 π 中的状态的集合。一条路径 $\pi = [s_i]_{i\geq 0}$ 是公平的，当且仅当对所有 $f \in F$ ，

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

计算与行为: 公平 Kripke 模型 K 上的一个计算即 Kripke 模型 $\langle S, R, I\rangle$ 上的一个计算。一个计算是公平的，当且仅当该计算的路径是公平的。 K 的公平计算的集合称为 K 的行为，记作 $[[K]]$ 。

公平状态: 一个状态是公平的，当且仅当存在从该状态出发的公平路径。

公平状态集合: 记 $\{s \mid s \to s', s' \in X\}$ 为 $R^{-1}(X)$ 。

设 $g : 2^S \to 2^S$ 。

定义 $g^0(X) = X$ 和 $g^{i+1}(X) = g^i(X)$ 。

若 $\forall k \geq 0.(g^k(\emptyset) \subseteq Y)$ 且 $\forall k \geq 0.(\bigvee_{i=0}^{k} .g^k(\emptyset) \subseteq Y')) \to (Y \subseteq Y')$ ，则称 Y 为 $g(Z)$ 的最小不动点集合，记作 $\mu Z.g(Z)$ 。 若 $\forall k \geq 0.(Y \subseteq g^k(S))$ 且 $\forall k \geq 0.(Y' \subseteq \bigwedge_{i=0}^{k} .g^k(S)) \to (Y' \subseteq Y)$ ，则称 ψ 为 $g(Z)$ 的最大不动点集合，记作 $\nu Z.g(Z)$ 。

定义 $S_F = \nu Z. \bigwedge_{f\in F} \mu Y.((f \cap R^{-1}(Z)) \cup R^{-1}(Y))$ 。

Proposition 2.9 *状态 $s \in S$ 是 K 的公平状态，当且仅当 $s \in S_F$ 。*

可达公平状态集合: 记由 A 可达的公平状态集合为 $rh_F(A)$ 。

Proposition 2.10 $rh_F(A) = rh(A) \cap S_F$ 。

M 的可达公平状态集合即由 I 可达的公平状态集合 $rh_F(I)$ 。

Corollary 2.3 $rh_F(I) = rh(I) \cap S_F$ 。

公平强连通分量: 考虑穷状态系统。给定 $K = \langle S, R, I, F\rangle$ 。有向图 (S, R) 的强连通分量也称为 K 的强连通分量。K 的强连通分量 e 是公平的，当且仅当 e 是非平凡的且 $\forall f \in F.(e \cap f \neq \emptyset)$ ，其中 $e \cap f$ 表示 f 中的元素和强连通分量 e 中的元素的交集。以下算法可检查 e 是否是公平的。

```
bool fairscc(K,e)
{   if (nontrivial(K,e)=false) return false;
    for (each f in F) if (e*f={}) { return false; }
    return true;
}
```

Proposition 2.11 *K 的强连通分量 e 是公平的，当且仅当 $fairscc(K, e)$ 为 $true$ 。*

公平状态判定: 在公平强连通分量检查算法的基础上, 以下算法可检查一个状态是否是公平的。

```
bool FairState(K,s)
{   G:=(S,R); scclist:=scctarjan(G);
    w:={}; for (each e in scclist) if (fairscc(K,e)) w=w+e;
    K'=(S,R,{s});
    return ReachbilityAnalysis(K',w);
}
```

Proposition 2.12 状态 $s \in S$ 是 K 的公平状态，当且仅当 $FairState(K, s)$ 为 $true$ 。

§2.2.1 公平安全性质与公平可达性质分析

Definition 2.7 (公平安全性质) 状态集合 $A \subseteq S$ 称为 K 的公平安全性质，当且仅当 K 的所有公平计算都是 A 计算。

Proposition 2.13 状态集合 $A \subseteq S$ 是 K 的公平安全性质当且仅当 $rh_F(I) \subseteq A$ 。

Definition 2.8 (公平可达性质) 状态集合 $A \subseteq S$ 称为 K 的公平可达性质，当且仅当 $[[K]]$ 中有能到达 A 状态的公平计算。

Proposition 2.14 状态集合 $A \subseteq S$ 是 K 的公平可达性质当且仅当 $rh_F(I) \cap A \neq \emptyset$ 。

Corollary 2.4 状态集合 $A \subseteq S$ 是 K 的公平安全性质，当且仅当状态集合 $\neg A$ 不是 K 的公平可达性质。

可达性分析: 给定公平 Kripke 模型 $K = \langle S, R, I, F\rangle$ 和集合 $A \subseteq S$ 。 A 是否是 K 的公平可达性质的分析可通过首先看哪些 A 状态是公平状态, 再结合可达性分析算法看这些公平状态中是否有由模型初始状态可达的。 根据原来的可达性分析算法修改的公平可达性分析算法如下。

```
bool FairReachability(K,A)
{   w:=I;
    repeat until w={};
        s:=w.getElement();
        if (s in A) and (FairState(K,s)=true) return true;
        visited[s]:=true;
        for each (s' in R(s)), if (visited[s']=false) w.putElement(s');
        w.removeElement(s);
    return false;
}
```

Proposition 2.15 状态集合 $A \subseteq S$ 是 K 的公平可达性质当且仅当 $FairReachability(K, A)$ 为 $true$ 。

Proposition 2.16 状态集合 $A \subseteq S$ 是 K 的公平安全性质当且仅当 $FairReachability(K, \neg A)$ 为 $false$ 。

由于 $FairReachability()$ 算法每次碰到一个 A 状态都要调用一次 $FairState()$ ， 我们可以考虑收集这些可达的 A 状态, 然后一次性地看其中是否有公平状态。以下算法可检查一个状态集合中是否有公平状态。设 B 为状态集合。

```
bool ExistFairState(K,B)
{   G:=(S,R); scclist:=scctarjan(G);
    w:={}; for (each e in scclist) if (fairscc(K,e)) w=w+e;
    K'=(S,R,B);
    return ReachbilityAnalysis(K',w);
}
```

Proposition 2.17 状态集合 $A \subseteq S$ 是 K 的公平可达性质, 当且仅当 $ExistFairState(K, rh(I) \cap A)$ 为 $true$ 。

Corollary 2.5 状态集合 $A \subseteq S$ 是 K 的公平安全性质, 当且仅当 $ExistFairState(K, rh(I) \cap \neg A)$ 为 $false$ 。

§2.2.2 公平必达性质与公平可免性质的分析

Definition 2.9 (公平必达性质) 状态集合 $A \subseteq S$ 称为 K 的公平必达性质，当且仅当 K 的所有公平计算都能到达 A 状态。

考虑有穷状态系统。用 $sc_F(B)$ 表示 $B \subseteq S$ 为公平强连通图，即 B 是非平凡强连通的且 $\forall f \in F.(B \cap f \neq \emptyset)$ 。

Proposition 2.18 状态集合 $A \subseteq S$ 是 K 的公平必达性质当且仅当 $\forall B \subseteq S.((B \cap I \neq \emptyset) \wedge \exists B' \subseteq B.(sc_F(B') \wedge \forall B'' \subseteq (B \setminus B').(B'' \neq \emptyset \rightarrow \exists s \in B''.(\exists s' \in (B \setminus B'').(s \rightarrow s')))) \rightarrow (B \cap A \neq \emptyset))$ 。

Definition 2.10 (公平可免性质) 状态集合 $A \subseteq S$ 称为 K 的公平可免性质，当且仅当 $[[K]]$ 中有公平 $\neg A$ 计算。

Proposition 2.19 状态集合 $A \subseteq S$ 是 K 的公平可免性质当且仅当 $\exists B \subseteq S.((B \cap I \neq \emptyset) \wedge \exists B' \subseteq B.(sc_F(B') \wedge \forall B'' \subseteq (B \setminus B').(\exists s \in B''.(B'' \neq \emptyset \rightarrow \exists s' \in (B \setminus B'').(s \rightarrow s')))) \wedge (B \cap A = \emptyset))$ 。

Corollary 2.6 状态集合 $A \subseteq S$ 是 K 的公平必达性质，当且仅当状态集合 A 不是 K 的公平可免性质。

公平可免性分析: 给定公平 Kripke 模型 $K = \langle S, R, I, F \rangle$ 和集合 $A \subseteq S$ 。 A 是否是 K 的公平可免性分析类似于原先的可免性分析，只是将所用到的非平凡强连通分量加强为公平强连通分量即可。定义 $\{f_1, ..., f_n\}|Y = \{f_1 \cap Y, ..., f_n \cap Y\}$ 。

```
bool FairAvoidability(K,A)
{   S':=S-A; R'=R|S'; I':=I-A; F'=F|S';
    G:=(S',R'); K':=(S',R',I'); K'':=(S',R',I',F');
    scclist:=scctarjan(G);
    w:={}; for (each e in scclist) if (fairscc(K'',e)) w:=(w+e);
    return ReachbilityAnalysis(K',w);
}
```

Proposition 2.20 状态集合 $A \subseteq S$ 是 K 的公平可免性质当且仅当 $FairAvoidability(K, A)$ 为 $true$ 。

Corollary 2.7 状态集合 $A \subseteq S$ 是 K 的公平必达性质当且仅当 $FairAvoidability(K, A)$ 为 $false$ 。

定义 $\langle S, R, I, F\rangle|X = \langle X, R|X, I \cap X, F|X\rangle$ 。

由于 A 是公平可免性质当且仅当 $K|\neg A$ 有公平可达状态，使用 $ExistFairState()$ ，则公平可免性分析可以更加简单。

Proposition 2.21 *$A \subseteq S$ 是 K 的公平可免性质当且仅当 $ExistFairState(K|\neg A, I \setminus A)$ 为 $true$ 。*

§2.2.3 模型非空问题

由于对计算增加了限制，有可能使得所有计算都成了非公平的计算，这样，模型的行为就是一个空集。 我们可以检查模型的行为是否为空。这个问题也称为模型非空问题。模型 K 非空记作 $K \neq \emptyset$ ，当且仅当 $[[K]] \neq \emptyset$ 。关于模型非空问题，我们有以下结论。

Proposition 2.22 *给定 $K = \langle S, R, I, F\rangle$ 。 $K \neq \emptyset$ 当且仅当存在由 I 可达的公平强连通分量。*

模型空性检查算法: 检查模型是否为空的算法称为空性检查算法。给定 $K = \langle S, R, I, F\rangle$ 。根据以上命题，模型的空性检查算法如下。

```
bool EmpChecking(K)
{   G:=(S,R);
    scclist:=scctarjan(G);
    for (each e in scclist) if (fairscc(K,e)) w=w+e;
    return ReachbilityAnalysis(K,w);
}
```

Proposition 2.23 *$K = \emptyset$ 当且仅当 $EmpChecking(K)$ 为 $true$ 。*

若 $|F| = 1$ ，即 F 中只有一个元素，则我们可以用一个双深度优先算法计算模型是否为空问题。给定 $K = \langle S, R, I, \{f\}\rangle$ 。模型的空性检查算法如下。

```
bool EmpCheckingDDFS(K)
{   w:=[]; a:=b:={};
    for each (s in I), if not (s in a)
            a.put(s);
            w.push(s); if (empcheckingDFS1(s)=false) return false; w.pop();
    return true;
}
bool empcheckingDFS1(v)
{   for each (s in R(v)), if not (s in a)
            a.put(s);
            w.push(s); if (empcheckingDFS1(s)=false) return false; w.pop();
    if (v in f) { b.put(s); if (empcheckingDFS2(s)=false) return false; }
    return true;
}
bool empcheckingDFS2(v)
{   for each (s in R(v)),
        if (s in w) return false;
```

```
            if not (s in b)
            { b.put(s); if (empcheckingDFS2(s)=false) return false; }
        return true;
    }
```

Proposition 2.24 给定 $K = \langle S, R, I, F\rangle$ 且 $|F| = 1$。$K = \emptyset$ 当且仅当 $EmpCheckingDDFS(K)$ 为 $true$。

非空问题的应用: 非空问题是一个基本问题，可应用于解决其它的问题。比如，可达性问题可以转换到非空问题。给定 Kripke 模型 $K = \langle S, R, I\rangle$ 和 $A \subseteq S$。定义 S', R', I', F 如下。

$$\begin{array}{|l}\hline S' = S \cup \{t\}\\ R' = R \cup \{(s,t) \mid s \in A\} \cup \{(t,t)\}\\ I' = I\\ F = \{\{t\}\}\\ \hline\end{array}$$

Proposition 2.25 状态集合 $A \subseteq S$ 是 K 的可达性质，当且仅当 $K' = \langle S', R', I', F\rangle$ 非空，当且仅当 $EmpCheckingDDFS(K')$ 为 $false$。

可免性问题可以转换到非空问题。给定 Kripke 模型 $K = \langle S, R, I\rangle$ 和 $A \subseteq S$。定义 S', R', I', F 如下。

$$\begin{array}{|l}\hline S' = S \cup \{t\}\\ R' = \{(s,s') \mid (s,s') \in R, s \notin A\} \cup \{(s,t) \mid s \in A\} \cup \{(t,t)\}\\ I' = I\\ F = \{S\}\\ \hline\end{array}$$

Proposition 2.26 状态集合 $A \subseteq S$ 是 K 的可免性质，当且仅当 $K' = \langle S', R', I', F\rangle$ 非空，当且仅当 $EmpCheckingDDFS(K')$ 为 $false$。

§2.2.4 强公平与弱公平条件

我们对公平 Kripke 模型，可以增强其公平条件。

强公平 Kripke 模型: 用 $F \subseteq 2^S \times 2^S$ 取代原来 $K = \langle S, R, I, F\rangle$ 的公平条件，并将公平路径定义如下。一条路径 $\pi = [s_i]_{i\geq 0}$ 是公平的，当且仅当对所有 $(f, g) \in F$，

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

按照这样的公平路径的定义，我们有一个强公平 Kripke 模型。

强公平 Kripke 模型增强了原公平 Kripke 模型的表达能力。

给定公平 Kripke 模型 $K = \langle S, R, I, \{g_1, ..., g_k\}\rangle$。定义 $F' = \{(S, g_1), ..., (S, g_k)\}$。则强公平 Kripke 模型 $K' = \langle S, R, I, F'\rangle$ 与 K 有相同的公平计算集合。

强公平 Kripke 模型对原公平 Kripke 模型的增强可以用如下例子作证。设 $K_1 = \langle S, R, I, F\rangle$ 其中 S, R, I, F 定义如下。

$$\begin{array}{|l}\hline S = \{s0, s1, s2\}\\ R = \{(s0,s0), (s1,s1), (s2,s2), (s0,s1), (s0,s2), (s1,s0), (s2,s0)\}\\ I = \{(s0,s0)\}\\ F = \{(\{s0\}, \{s0\}), (\{s0\}, \{s0\})\}\\ \hline\end{array}$$

容易验证没有公平 Kripke 模型能够有与强公平 Kripke 模型 K_1 相同的公平计算的集合。

弱公平 Kripke 模型: 弱公平 Kripke 模型的形式与强公平 Kripke 模型相同, 即用 $F \subseteq 2^S \times 2^S$ 取代原来 $K = \langle S, R, I, F\rangle$ 的公平条件。不同的是对公平路径的定义。在这里, 一条路径 $\pi = [s_i]_{i\geq 0}$ 是公平的，当且仅当对所有 $(f, g) \in F$,

$$inf(\pi) \subseteq f \to inf(\pi) \cap g \neq \emptyset$$

按照这样的公平路径的定义，我们有一个弱公平 Kripke 模型。

弱公平 Kripke 模型在一些情况下与原公平 Kripke 模型相比较有表达的方便性，但与原公平 Kripke 模型的表达能力相同。

给定公平 Kripke 模型 $K = \langle S, R, I, \{g_1, ..., g_k\}\rangle$ 。定义 $F' = \{(S, g_1), ..., (S, g_k)\}$ 。则弱公平 Kripke 模型 $K' = \langle S, R, I, F'\rangle$ 与 K 有相同的公平计算集合。

给定弱公平 Kripke 模型 $K = \langle S, R, I, \{(f_1, g_1), ..., (f_k, g_k)\}\rangle$ 。定义 $F' = \{g_1 \cup (S \setminus f_1), ..., g_k \cup (S \setminus f_k)\}$ 。则公平 Kripke 模型 $K' = \langle S, R, I, F'\rangle$ 与 K 有相同的公平计算集合。

§2.3 标号 Kripke 模型

Kripke 模型对于系统的描述过于简化，只使用状态集表示性质等。为了表达和计算的简便，可用命题表示性质。我们可将结构中的状态和命题建立联系，使得我们能知道每个状态上哪些命题是成立的。

Definition 2.11 *给定原子命题的有穷集合 AP 。一个 AP 上的标号 Kripke 模型是一个四元组 $\langle S, R, I, L\rangle$ 其中 $\langle S, R, I\rangle$ 为 Kripke 模型， $L : S \to 2^{AP}$ 为状态集合到 AP 的幂集的映射。*

$L(s)$ 表示在 s 上成立的所有原子命题组成的集合，即原子命题 p 在 s 上成立当且仅当 $p \in L(s)$ 。

假定原子命题集合 AP 和 AP 上的标号 Kripke 模型 $K = \langle S, R, I, L\rangle$ 为给定。

记原子命题集合 AP 上所有命题公式的集合为 $\mathcal{L}_{AP}$ 。

可达状态: 标号 Kripke 模型 K 的可达状态即 $K' = \langle S, R, I\rangle$ 的可达状态。

计算与行为: 标号 Kripke 模型 K 上的路径即 $K' = \langle S, R, I\rangle$ 上的路径。 K 的计算即 K' 的计算，其行为 $[[K]] = [[K']]$ 。

状态与公式: 我们用命题逻辑的公式表示状态的性质。状态 $s \in S$ 满足 $\varphi \in \mathcal{L}_{AP}$ 记作 $K, s \models \varphi$, 定义如下。

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

定义 $K, X \models \varphi$, 当且仅当对所有 $s \in X$ 都有 $K, s \models \varphi$ 。

设 $m = \{x_1, ..., x_k\} \subseteq AP$ 且 $\{y_1, ..., y_l\} = AP \setminus m$ 。

定义 $m \models \varphi$ 当且仅当 $(\varphi^{1,...,1}_{x_1,...,x_k})^{0,...,0}_{y_1,...,y_l}$ 的值为 1 。

为方便起见，在 K 为给定且不引起混淆的情况下， $K, s \models \varphi$ 和 $K, X \models \varphi$ 亦可分别写做 $s \models \varphi$ 和 $X \models \varphi$ 。

Proposition 2.27 *$s \models \varphi$ 当且仅当 $L(s) \models \varphi$ 。*

状态集合与公式的对应: 定义 $[[s]] = (\bigwedge_{p\in L(s)} p) \wedge (\bigwedge_{p\in AP\setminus L(s)} \neg p)$ 。定义 $[[X]] = \bigvee_{s\in X}[[s]]$ 。

Proposition 2.28 *$X \models \varphi$ 当且仅当 $[[X]] \to \varphi$ 。*

Proposition 2.29 *若 $X \subseteq Y$ 则 $[[X]] \to [[Y]]$ 。*

Proposition 2.30 *若 L 为单射函数，则以下成立。 $[[X]] \to [[Y]]$ 则 $X \subseteq Y$ 。*

公式与状态集合的对应: 定义 $[[\varphi]] = \{s \mid s \models \varphi\}$ 。 $[[\varphi]]$ 即满足 φ 的状态的集合。 称满足 φ 的状态为 φ 状态。 若一条路径上的所有状态都满足 φ ，则称该路径为 φ 路径。若一个计算上的所有状态都满足 φ ，则称该计算为 φ 计算。

Proposition 2.31 *$\varphi \to \psi$ 当且仅当 $[[\varphi]] \subseteq [[\psi]]$ 。*

Proposition 2.32 *$\varphi \leftrightarrow [[([[\varphi]])]]$ 。*

Proposition 2.33 *若 L 为单射函数，则 $X = [[([[X]])]]$ 。*

正确性性质: 我们可将原来用状态集合定义的可达性质、安全性质、可免性质、必达性质重新用公式定义。

- φ 是 $K = \langle S, R, I, L\rangle$ 的可达性质，当且仅当 $[[\varphi]]$ 是 $K' = \langle S, R, I\rangle$ 的可达性质，即 $rh(I) \cap [[\varphi]] \neq \emptyset$ ，即 $\exists s \in rh(I).(s \models \varphi)$ 。
- φ 是 $K = \langle S, R, I, L\rangle$ 的安全性质，当且仅当 $[[\varphi]]$ 是 $K' = \langle S, R, I\rangle$ 的安全性质，即 $rh(I) \subseteq [[\varphi]]$ ， 即 $rh(I) \models \varphi$ 。
- φ 是 $K = \langle S, R, I, L\rangle$ 的可免性质，当且仅当 $[[\varphi]]$ 是 $K' = \langle S, R, I\rangle$ 的可免性质，即 $\exists B \subseteq S.((B \cap rh(I) \neq \emptyset) \wedge (\forall s \in B.(\exists s' \in B.(s \to s')) \wedge (\forall s \in B.(s \not\models \varphi)))$ 。
- φ 是 $K = \langle S, R, I, L\rangle$ 的必达性质，当且仅当 $[[\varphi]]$ 是 $K' = \langle S, R, I\rangle$ 的必达性质， 即 $\forall B \subseteq S.((B \cap rh(I) \neq \emptyset) \wedge (\forall s \in B.(\exists s' \in B.(s \to s'))) \to (\exists s \in B.(s \models \varphi)))$ 。

分析方法: 标号 Kripke 模型在原来 Kripke 模型之上建立了状态与命题逻辑公式的联系，原有的概念和分析方法可以经过相应改变后使用。以下仅讨论安全性质与必达性质的推理问题。

安全性质的推理: 定义 $R(\varphi) = [[\{s' \mid s \to s', s \models \varphi\}]]$ 。若 $R(\varphi) \to \varphi$ ，则称 φ 为迁移不变量。关于安全性质，我们有如下推理规则。设 φ', φ 为公式。

$$\frac{\begin{array}{l}[[I]] \to \varphi' \\ \varphi' \text{ 是迁移不变量} \\ \varphi' \to \varphi\end{array}}{\varphi \text{ 是安全性质}}$$

可靠性与完备性: 以上推理规则是可靠的且若 L 为单射函数则推理规则是完备的。假定前提成立，由前两条我们知道 $rh(I) \models \varphi'$ ，又由第三条我们知道 $rh(I) \models \varphi$ ，所以 φ 是安全性质，因而推理规则是可靠的。假定 φ 是安全的，取 $\varphi' = [[rh(I)]]$ ，由 L 为单射函数知 $s \models \varphi'$ 则 $s \in rh(I)$ ，所以前提条件成立，由此可证 φ 是安全性质，因而推理规则是完备的。

必达性质的推理: 关于必达性质，我们有如下推理规则。设 $\psi_0, \psi_1, ..., \psi_n, \varphi$ 为公式。

$$\frac{\begin{array}{l}[[I]] \to \psi_0 \\ \forall i \in \{0, ..., n-1\}.(R(\psi_i \wedge \neg\varphi) \to \psi_{i+1}) \\ \psi_n \to \varphi\end{array}}{\varphi \text{ 是必达性质}}$$

可靠性与完备性: 以上推理规则是可靠的且对于有穷状态系统若 L 是单射函数则推理规则是完备的。

§2.4 公平标号 Kripke 模型

我们可以结合标号 Kripke 模型和公平 Kripke 模型构造标号公平 Kripke 模型，这样我们可以使用公式描述性质，也可以对合理的计算设置条件。我们可以在公平 Kripke 模型加标号也可以在标号 Kripke 模型加公平条件。 本节的选择是在标号 Kripke 模型上加公平条件。

Definition 2.12 给定一个有穷命题集合 AP。一个 AP 上的公平标号 *Kripke* 模型是一个五元组 $\langle S, R, I, L, F\rangle$ 其中 $\langle S, R, I, L\rangle$ 是一个 AP 上的标号 *Kripke* 模型， $F \subseteq \mathcal{L}_{AP}$ 为公平性约束的有穷集合。

计算与行为: 公平标号 Kripke 模型 $K = \langle S, R, I, L, F\rangle$ 的计算就是 Kripke 模型 $\langle S, R, I\rangle$ 的计算， 设 $F = \{\phi_1, ..., \phi_k\}$。 K 的公平计算就是公平 Kripke 模型 $K' = \langle S, R, I, \{[[\phi_1]], ..., [[\phi_k]]\}\rangle$ 的公平计算，其行为 $[[K]] = [[K']]$。

正确性性质及其分析方法: 由公式与状态集合的对应关系，我们可以相应地定义公平标号 Kripke 模型的公平安全和公平必达等性质，并将标号 Kripke 模型和公平 Kripke 模型的概念和分析方法经过相应改变后使用。以下仅讨论语言及语言非空问题。

模型语言: 给定一个计算，若我们将每个状态替换成这个状态所标的命题集合，我们将得到一个命题集合的无穷串。与公平计算对应的这些无穷串的集合称为模型的语言，定义如下。设 $\pi = [\pi_i]_{i\geq 0}$。定义 $L(\pi) = [L(\pi_i)]_{i\geq 0}$。集合 $\{L(\pi) \subseteq (2^{AP})^\omega \mid \pi \in [[K]]\}$ 称为 K 的语言，记作 $\mathcal{L}(K)$。

语言非空问题: 语言非空问题可转化成模型非空问题。我们有以下结论。

Proposition 2.34 给定 $K = \langle S, R, I, L, F\rangle$ 且 $F = \{\phi_1, ..., \phi_k\}$。 $\mathcal{L}(K) \neq \emptyset$，当且仅当 $[[K]]$ 非空，当且仅当公平 *Kripke* 模型 $K' = \langle S, R, I, \{[[\phi_1]], ..., [[\phi_k]]\}\rangle$ 非空，即 $EmpChecking(K')$ 为 $false$。

§2.5 练习

1. 给定公平 Kripke 模型 $K = \langle S, R, I, F\rangle$ 和 $A \subseteq S$。用模型非空问题算法检查 A 是否是 K 的可达性质。
2. 给定公平 Kripke 模型 $K = \langle S, R, I, F\rangle$ 和 $A \subseteq S$。用模型非空问题算法检查 A 是否是 K 的可免性质。

§3 变量迁移模型

本章介绍基于一阶逻辑的以变量的赋值为特点的模型。 这类模型将状态分解为不同的分量，状态的迁移基于变量的重新赋值。部分模型亦包含控制状态及控制状态的转移规则。

基本符号: 给定常元和函数符号集合 F ，谓词符号集合 P 。

$B = (F, P)$ 上一阶逻辑公式集合记为 WFF_B 。

B 上的项的集合记为 T_B 。

WFF_B 中不带量词的公式集合记为 QFF_B 。

B 的解释确定了迁移模型描述中常元符号、函数符号和谓词符号的含义。

给定 B 的解释 I 。一个赋值 σ 满足公式 φ 当且仅当 $I(\varphi)(\sigma) = true$ ，记作 $\sigma \models_I \varphi$ 。 所有赋值都满足公式 φ ，记作 $\models_I \varphi$ 。

变量限制在 V 中的项的集合记作 $T_{(B,V)}$ 。

自由变量限制在 V 中的公式的集合记作 $WFF_{(B,V)}$ 。

自由变量限制在 V 中的 QFF_B 公式的集合记作 $QFF_{(B,V)}$ 。

以下假定 $B = (F, P)$ 以及 B 的解释 $I = (D, I_0)$ 为给定。

假定 P 中包含命题常量 $true$ 和 $false$ 。为方便起见，用 $\varphi = \emptyset$ 表示 $\models_I (\varphi \leftrightarrow false)$ 。

不动点公式: 设 $g : WFF_B \to WFF_B$ 。 定义 $g^0(\varphi) = \varphi$ 和 $g^{i+1}(\varphi) = g^i(\varphi)$ 。

若 $\forall k \geq 0.(g^k(false) \to \psi)$ 且 $\forall \psi'.(\forall k \geq 0.(g^k(false) \to \psi') \to (\psi \to \psi'))$ ，则称 ψ 为 $g(Z)$ 的最小不动点公式，记作 $\mu Z.g(Z)$ 。若 $\forall k \geq 0.(\psi \to g^k(true))$ 且 $\forall \psi'.(\forall k \geq 0.(\psi' \to g^k(true)) \to (\psi' \to \psi))$ ，则称 ψ 为 $g(Z)$ 的最大不动点公式，记作 $\nu Z.g(Z)$ 。

称一个不动点公式为良定义的，当且仅当存在一个具有所述相关性质的公式。

若对任意公式 φ, ψ 都有 $\varphi \to \psi$ 蕴涵 $g(\varphi) \to g(\psi)$ ，则称函数 g 为单调函数。

以下假定所有单调函数的不动点公式都是良定义的。

§3.1 卫式迁移模型

卫式迁移模型是一阶逻辑的扩充。 卫式迁移模型所描述的迁移关系主要有两个要素：卫式和赋值。除此之外，卫式迁移模型包含初始条件的描述。 在一阶逻辑的基础上，增加以下符号集合：

$$\{\longrightarrow, :=\}$$

给定变量集合 V 。定义 (B, V) 上的迁移如下。

$$\varphi \longrightarrow (x_1, x_2, ..., x_k) := (e_1, e_2, ..., e_k)$$

其中 $\varphi \in QFF_{(B,V)}$ ， $\{x_1, ..., x_k\} \subseteq V$ 且各不相同， $\{e_1, ..., e_k\} \subseteq T_{(B,V)}$ 。 一个迁移表示模型的一个原子动作。

Definition 3.1 *一个 (B, V) 上的卫式迁移模型是一个二元组 (T, Θ) 其中 T 为 (B, V) 上迁移的有穷集合， $\Theta \in QFF_{(B,V)}$ 为初始条件。*

以下假定卫式迁移模型 $M = (T, \Theta)$ 为给定。

状态与状态空间: 变量的赋值 σ 代表了计算过程中变量在某一时刻的状态。因此称 σ 为状态。模型的状态空间为 V 中变量取值的组合 $\Sigma = \{\sigma \mid \sigma(x) \in D, x \in V\}$ 。状态的有效部分取决于 V 中变量的取值。若 $V = \{v_1, ..., v_n\}$ ，则 σ 亦可表示为 n 元组 $\langle \sigma(v_1), ..., \sigma(v_n) \rangle$ 。 若 $\sigma \models_I \varphi$ ，则称 σ 为 φ 状态。 φ 状态的集合记为 $[[\varphi]]$ 。

状态迁移关系: 设 t 是 $\varphi \longrightarrow (x_1, x_2, ..., x_k) := (e_1, e_2, ..., e_k)$ 为一个迁移。 t 在状态 σ 可执行当且仅当 $\sigma \models_I \varphi$ 。 $\sigma \xrightarrow{t} \sigma'$ 当且仅当以下成立。

t 在状态 σ 可执行且
$\sigma' = \sigma[x_1/I(e_1)(\sigma)] \cdots [x_k/I(e_k)(\sigma)]$ 。

模型的迁移关系 $\rightarrow$ 为 Σ^2 的一个子集。 $\sigma \rightarrow \sigma'$ 当且仅当以下一种情况成立。

在状态 σ ，存在可执行迁移 t 且 $\sigma \xrightarrow{t} \sigma'$
在状态 σ ，不存在可执行迁移且 $\sigma' = \sigma$

可达状态: 记由 φ 状态可达的状态集合为 $rh(\varphi)$ 。我们有 $rh(\varphi) = \{\sigma' \mid \sigma \xrightarrow{*} \sigma', \sigma \models_I \varphi\}$ 。 M 的可达状态集合即由 Θ 状态可达的状态集合 $rh(\Theta) = \{\sigma' \mid \sigma \xrightarrow{*} \sigma', \sigma \models_I \Theta\}$ 。

路径: 无穷序列 $[\sigma_i]_{i \geq 0}$ 称为 M 的路径，当且仅当 $[\sigma_i]_{i \geq 0}$ 是状态集 Σ 上的一个无穷序列且对任意 $i \geq 0$ 都有 $\sigma_i \rightarrow \sigma_{i+1}$ 。

计算与行为: 无穷序列 $[\sigma_i]_{i \geq 0}$ 称为 M 的计算，当且仅当 $[\sigma_i]_{i \geq 0}$ 是 M 的路径且 $\sigma_0 \models_I \Theta$ 。 M 的计算的集合称为 M 的行为，记作 $[[M]]$ 。

§3.1.1 正确性性质

Definition 3.2 (安全性质) φ 是 M 的安全性质，当且仅当 M 的所有计算都是 φ 计算。

设 $T = \{p_i \rightarrow e_i \mid i = 1, ..., k\}$ 。
记 $\bigvee_{i=1}^{k} p_i$ 为 $E(T)$ 。
记 $T \cup \{E(T) \rightarrow (x_1) := (x_1)\}$ 为 T^+ 。

Definition 3.3 设 t 为 $p \longrightarrow (x_1, x_2, ..., x_k) := (e_1, e_2, ..., e_k)$ 。定义 $[W]\psi$ 如下。

$$\begin{aligned} [\{\}]\psi &= true \\ [\{t\}]\psi &= p \rightarrow \psi(e_1/x_1, ..., e_n/x_n) \\ [X \cup Y]\psi &= [X]\psi \wedge [Y]\psi \end{aligned}$$

Proposition 3.1 $\varphi \equiv [T^+]\psi$ 当且仅当 $\forall \sigma.(I(\varphi)\sigma \rightarrow \forall \sigma'.((\sigma \rightarrow \sigma') \rightarrow I(\psi)\sigma'))$ 。

定义 $\langle Y \rangle \varphi = \neg[Y]\neg\varphi$ 。

Proposition 3.2 φ 是 M 的安全性质，当且仅当 $\forall k \geq 0.(\Theta \wedge (\langle T^+ \rangle)^k \neg\varphi = \emptyset)$ 。

Definition 3.4 (必达性质) φ 是 M 的必达性质，当且仅当 M 的所有计算都能到达 φ 状态。

Proposition 3.3 φ 是 M 的必达性质当且仅当 $\forall \psi.((\psi \wedge \Theta \neq \emptyset) \wedge (\psi \rightarrow \langle T^+ \rangle \psi) \rightarrow (\psi \wedge \varphi \neq \emptyset))$ 。

§3.1.2 卫式迁移模型与标号 Kripke 模型的等价

给定 (B, V) 上的卫式迁移模型 $M = (T, \Theta)$ 。给定 $AP = \{p_1, ..., p_n\}$ 上的标号 Kripke 模型 $K = \langle S, R, I, L \rangle$ 。

设 $BP \subseteq QFF_{(B,V)}$ 。

定义 $\mathcal{L}_{BP}$ 为以 BP 中的命题为原子命题的布尔公式。

设 $\zeta : AP \rightarrow BP$ 是一一对应关系。

定义 $\zeta^{-1}(\varphi)$ 为对所有 φ 中出现的 BP 中的公式用 ζ 关系中对应的 AP 中的命题替换得到的结果。

Definition 3.5 M 与 K 是 ζ 计算等价的，当且仅当对任意 M 的计算 $[\sigma_i]_{i\geq 0}$ 存在 K 的计算 $[s_i]_{i\geq 0}$ 满足 $\forall i \geq 0. \forall \psi \in BP. (\sigma_i \models \psi \leftrightarrow s_i \models \zeta^{-1}(\psi))$ ，反之亦然。

Proposition 3.4 设 M 与 K 是 ζ 计算等价的。设 $\varphi \in \mathcal{L}_{BP}$ 。 φ 是 M 的安全性质当且仅当 $\zeta^{-1}(\varphi)$ 是 K 的安全性质； φ 是 M 的必达性质当且仅当 $\zeta^{-1}(\varphi)$ 是 K 的必达性质。

卫式迁移模型的标号 Kripke 模型的构造: 我们可以构造与卫式迁移模型等价的标号 Kripke 模型， 用标号 Kripke 模型性质的分析方法来分析卫式迁移模型的性质。

给定 (B, V) 上的卫式迁移模型 $M = (T, \Theta)$ 其中 $V = \{v_1, ..., v_n\}$ 。

定义

$S = D^n$ 。
$R = \{(\langle\sigma(v_1), ...\sigma(v_n)\rangle, \langle\sigma'(v_1), ...\sigma'(v_n)\rangle) \mid \sigma \rightarrow \sigma'\}$ 。
$I = \{\langle\sigma(v_1), ...\sigma(v_n)\rangle \mid \sigma \models_I \Theta\}$ 。

设 $BP = \{\psi_1, ..., \psi_n\} \subseteq QFF_{(B,V)}$ 。

定义 $AP = \{p_1, ..., p_n\}$ 且定义 $L : S \rightarrow 2^{AP}$ 如下： $p_i \in L(\langle\sigma(v_1), ...\sigma(v_n)\rangle) \leftrightarrow \sigma \models \psi_i$ 。

定义 $\zeta : AP \rightarrow BP$ 如下： $\zeta(p_1) = \psi_1, ..., \zeta(p_n) = \psi_n$ 。

定义 $(M)^{km} = \langle S, R, I, L\rangle$ 。

Proposition 3.5 M 与 $(M)^{km}$ 是 ζ 计算等价的。

Corollary 3.1 设 $\varphi \in \mathcal{L}_{BP}$ 。 φ 是 M 的安全性质当且仅当 $\zeta^{-1}(\varphi)$ 是 $(M)^{km}$ 的安全性质； φ 是 M 的必达性质当且仅当 $\zeta^{-1}(\varphi)$ 是 $(M)^{km}$ 的必达性质。

§3.2 公平卫式迁移模型

我们可以在卫式迁移模型的基础上设置公平计算的条件，定义公平卫式迁移模型。

给定变量集合 V 。

Definition 3.6 一个 (B, V) 上的公平卫式迁移模型是一个三元组 (T, Θ, Φ) 其中 (T, Θ) 为 (B, V) 上的卫式迁移模型， $\Phi \subseteq QFF_{(B,V)}$ 为公平条件，是一个自由变量在 V 中且不带量词的公式的有穷集合。

以下假定公平卫式迁移模型 $M = (T, \Theta, \Phi)$ 为给定。

状态空间与迁移关系: M 的状态空间等同于 (T, Θ) 的状态空间。 M 的状态迁移关系等同于 (T, Θ) 的状态迁移关系。

公平路径: 无穷序列 $[\sigma_i]_{i\geq 0}$ 称为 M 的路径，当且仅当 $[\sigma_i]_{i\geq 0}$ 是 (T, Θ) 的路径。无穷序列 $[\sigma_i]_{i\geq 0}$ 称为 M 的公平路径，当且仅当 $[\sigma_i]_{i\geq 0}$ 是 M 的路径且对所有 $\phi \in \Phi$ 都有无限多个 i 使得 $\sigma_i \models_I \phi$ 。

公平计算与行为: 无穷序列 $[\sigma_i]_{i\geq 0}$ 称为 M 的计算，当且仅当 $[\sigma_i]_{i\geq 0}$ 是 (T, Θ) 的计算。无穷序列 $[\sigma_i]_{i\geq 0}$ 称为 M 的公平计算，当且仅当 $[\sigma_i]_{i\geq 0}$ 是 M 的计算且 $[\sigma_i]_{i\geq 0}$ 是 M 的公平路径。 M 的公平计算的集合称为 M 的行为，记作 $[[M]]$ 。

公平状态: 一个状态是公平状态，当且仅当存在一条从该状态出发的公平路径。 定义 $\Sigma_F = \nu Z.\bigwedge_{f\in F}\mu Y.((f \wedge \langle T^+\rangle(Z)) \vee \langle T^+\rangle(Y))$ 。

Proposition 3.6 σ 是公平状态当且仅当 $\sigma \models_I \Sigma_F$ 。

§3.2.1 正确性性质

Definition 3.7 (公平安全性质) φ 是 M 的公平安全性质，当且仅当 M 的所有公平计算都是 φ 计算。

Proposition 3.7 φ 是 M 的公平安全性质，当且仅当 $\Theta \wedge \mu Y.((\neg\varphi \wedge \Sigma_F) \vee \langle T^+\rangle(Y)) = \emptyset$ 。

Definition 3.8 (公平必达性质) φ 是 M 的公平必达性质，当且仅当 M 的所有公平计算都能到达 φ 状态。

Proposition 3.8 ψ 是 K 的公平必达性质当且仅当 $\Theta \wedge \nu Z.(\neg\varphi \wedge \bigwedge_{f\in F}\langle T^+\rangle(\mu Y.((Z \wedge f) \vee (\neg\varphi \wedge \langle T^+\rangle(Y))))) = \emptyset$ 。

§3.2.2 公平卫式迁移模型与公平标号 Kripke 模型的等价

给定 (B, V) 上的公平卫式迁移模型 $M = (T, \Theta, \Phi)$ 。给定 $AP = \{p_1, ..., p_n\}$ 上的公平标号 Kripke 模型 $K = \langle S, R, I, L, F\rangle$ 。

设 $BP \subseteq QFF_{(B,V)}$ 。

设 $\zeta : AP \to BP$ 是一一对应关系。

Definition 3.9 M 与 K 是 ζ 计算等价的，当且仅当对任意 M 的公平计算 $[\sigma_i]_{i\geq 0}$ 存在 K 的公平计算 $[s_i]_{i\geq 0}$ 满足 $\forall i \geq 0.\forall \psi \in BP.(\sigma_i \models \psi \leftrightarrow s_i \models \zeta^{-1}(\psi))$ ，反之亦然。

Proposition 3.9 设 M 与 K 是 ζ 计算等价的。设 $\varphi \in \mathcal{L}_{BP}$ 。 φ 是 M 的公平安全性质当且仅当 $\zeta^{-1}(\varphi)$ 是 K 的公平安全性质； φ 是 M 的公平必达性质当且仅当 $\zeta^{-1}(\varphi)$ 是 K 的公平必达性质。

公平卫式迁移模型的公平标号 Kripke 模型的构造: 类似于卫式迁移模型，我们可以构造与公平卫式迁移模型等价的公平标号 Kripke 模型，用公平标号 Kripke 模型性质的分析方法来分析公平卫式迁移模型的性质。

给定 (B, V) 上的卫式迁移模型 $M = (T, \Theta, \Phi)$ 其中 $V = \{v_1, ..., v_n\}$ 。

设 $BP = \{\psi_1, ..., \psi_n\} \subseteq QFF_{(B,V)}$ 且 $\forall \phi \in \Phi.(\phi \in \mathcal{L}_{BP})$ 。

$\langle S, R, I\rangle$ 的定义同其在 §3.1.2 中的定义。

定义 $AP = \{p_1, ..., p_n\}$ 且定义 $L : S \to 2^{AP}$ 如下： $p_i \in L(\langle\sigma(v_1), ..., \sigma(v_n)\rangle) \leftrightarrow \sigma \models \psi_i$ 。

定义 $\zeta : AP \to BP$ 如下： $\zeta(p_1) = \psi_1, ..., \zeta(p_n) = \psi_n$ 。

定义 $F = \{\zeta^{-1}(\phi) \mid \phi \in \Phi\}$ 。

定义 $(M)^{fkm} = \langle S, R, I, L, F\rangle$ 。

Proposition 3.10 M 与 $(M)^{fkm}$ 是 ζ 计算等价的。

Corollary 3.2 设 $\varphi \in \mathcal{L}_{BP}$ 。 φ 是 M 的公平安全性质当且仅当 $\zeta^{-1}(\varphi)$ 是 $(M)^{fkm}$ 的公平安全性质； φ 是 M 的公平必达性质当且仅当 $\zeta^{-1}(\varphi)$ 是 $(M)^{fkm}$ 的公平必达性质。

§3.3 流程图模型

流程图模型 是一阶逻辑的扩充。 流程图模型主要有两种语句。一种是赋值语句。一种是条件语句。这些语句称为指令。一个指令表示模型的一个原子动作。 在一阶逻辑的基础上，我们增加以下两个集合的符号：

$$\text{辅助符号集 } H = \{:=, :, goto, if, else, fi\}$$
$$\text{标号符号集 } LB = \{beg, end, l_1, l_2, ...,\}$$

给定变量集合 V 。定义 (B, V) 上的流程图模型的两种指令如下。

l_1: $(x_1, ..., x_k) := (e_1, ..., e_k)$ goto l_2
l_1: if (b) goto l_2 else goto l_3 fi

其中 $l_1, l_2, l_3 \in LB$ 且 $l_1 \neq end$ ， $\{x_1, ..., x_k\} \subseteq V$ 且各不相同， $\{e_1, ..., e_k\} \subseteq T_{(B,V)}$, $b \in QFF_{(B,V)}$ 。 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 3.10 一个 (B, V) 上的流程图模型 $M = (T)$ 其中 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

以下假定流程图模型 $M = (T)$ 为给定。

状态与状态空间: 流程图模型的状态有两个部分：即标号和变量状态。标号可以理解为模型的控制状态。变量状态空间为 V 中变量取值的组合 $\Sigma = \{\sigma \mid \sigma(x) \in D, x \in V\}$ 。 模型的状态空间为标号和 V 中变量取值的组合 $LB \times \Sigma$ 。

状态迁移关系: 给定指令 t 。 $(l_i, \sigma_i) \xrightarrow{t} (l_{i+1}, \sigma_{i+1})$ 当且仅当以下一种情况成立。

t 为 l_i: $(x_1, ..., x_k) := (e_1, ..., e_k)$ goto l_{i+1} ，
且 $\sigma_{i+1} = \sigma_i[x_1/I(e_1)(\sigma_i)] \cdots [x_k/I(e_k)(\sigma_i)]$
t 为 l_i: (if (b) goto l' else goto l'' fi ，
$\sigma_{i+1} = \sigma_i$ 且 $\sigma_i \models_I b$ 则 $l_{i+1} = l'$ ， 否则 $l_{i+1} = l''$

模型的迁移关系 $\rightarrow$ 为 $(LB \times \Sigma)^2$ 的一个子集。 $(l_i, \sigma_i) \rightarrow (l_{i+1}, \sigma_{i+1})$ 当且仅当以下一种情况成立。

存在指令 $t \in T$ 使得 $(l_i, \sigma_i) \xrightarrow{t} (l_{i+1}, \sigma_{i+1})$ 或
不存在定义标号 l_i 的指令且 $(l_i, \sigma_i) = (l_{i+1}, \sigma_{i+1})$ 。

流程图模型的迁移有如下特点。

Proposition 3.11 若 $(l, \sigma) \rightarrow (l', \sigma')$ 且 $(l, \sigma) \rightarrow (l'', \sigma'')$ 则 $(l', \sigma') = (l'', \sigma'')$ 。

计算: M 的一个计算是状态集 $LB \times \Sigma$ 上的一个无穷序列 $[(l_i, \sigma_i)]_{i \geq 0}$ 满足 $l_0 = beg$ 且对任意 $i \geq 0$ 都有 $(l_i, \sigma_i) \rightarrow (l_{i+1}, \sigma_{i+1})$ 。

可达状态： M 在初始变量状态为 σ 时可到达的状态的集合为 $\{(l,\sigma') \mid (beg,\sigma)\xrightarrow{*}(l,\sigma')\}$ 。 M 的可达状态集合为 $\{(l,\sigma') \mid (beg,\sigma)\xrightarrow{*}(l,\sigma'),\sigma\in\Sigma\}$ 。

终止状态： M 在初始变量状态为 σ 时计算能够终止，当且仅当存在 σ' 使得 $(beg,\sigma)\xrightarrow{*}(end,\sigma')$ 。若存在 σ 使得 $(beg,\sigma)\xrightarrow{*}(end,\sigma')$ ，则称 σ' 为 M 的一个终止状态。

计算结果的唯一性：流程图模型的计算结果有唯一性。

Corollary 3.3 *给定流程图模型 M 和状态 σ 。若 $(beg,\sigma)\xrightarrow{*}(end,\sigma')$ 且 $(beg,\sigma)\xrightarrow{*}(end,\sigma'')$ 则 $\sigma'=\sigma''$ 。*

正确性性质：设 $A\subseteq LB\times\Sigma$ 。

- A 是 M 的安全性质，当且仅当 M 的所有计算都是 A 计算。
- A 是 M 的必达性质，当且仅当 M 的所有计算都能到达 A 状态。

§3.3.1 正确性问题

流程图模型的基本正确性问题包括模型计算的终止性以及模型的终止状态和初始状态的关系。 M 的前置条件是对 M 的初始状态的变量状态的限制条件。 M 的后置条件是对 M 的终止状态的变量状态的限制条件。前置条件和后置条件可以是谓词或公式。设 φ 和 ψ 是谓词。

- 如果模型初始时的变量状态满足条件 φ ，给定 ψ ，如果计算终止，则计算终止时的变量状态满足 ψ 。这个要求称为模型对于前置条件 φ 和后置条件 ψ 的部分正确性。
- 如果模型初始时的变量状态满足条件 φ ，给定 ψ ，则要求计算能够终止且计算终止时的变量状态满足 ψ 。这个要求称为模型对于前置条件 φ 和后置条件 ψ 的完全正确性。

完全正确性包括了模型的终止性，即如果模型初始时的变量状态满足条件 φ ，则要求计算能够终止。

终止性：设 φ 是谓词。 M 对于前置条件 φ 的终止性定义如下。

$$\text{终止性：}\quad \forall\sigma.(\varphi(\sigma)\to\exists\sigma'.((beg,\sigma)\xrightarrow{*}(end,\sigma')))$$

Proposition 3.12 *M 对于前置条件 $true$ 是终止的，当且仅当 $\{(end,\sigma)\mid\sigma\in\Sigma\}$ 是 M 的必达性质。*

部分正确性：设 φ 和 ψ 是谓词。 M 对于前置条件 φ 和后置条件 ψ 的部分正确性定义如下。

$$\text{部分正确性：}\quad \forall\sigma.(\varphi(\sigma)\to\forall\sigma'.(((beg,\sigma)\xrightarrow{*}(end,\sigma'))\to\psi(\sigma')))$$

Proposition 3.13 *M 对于前置条件 $true$ 和后置条件 ψ 是部分正确的，当且仅当 $\{(end,\sigma)|\psi(\sigma)\}\cup\{(l,\sigma)\mid l\neq end\}$ 是 M 的安全性质。*

完全正确性：设 φ 和 ψ 是谓词。 M 对于前置条件 φ 和后置条件 ψ 的完全正确性定义如下。

$$\text{完全正确性：}\quad \forall\sigma.(\varphi(\sigma)\to\exists\sigma'.(((beg,\sigma)\xrightarrow{*}(end,\sigma'))\wedge\psi(\sigma')))$$

Proposition 3.14 *M 对于前置条件 φ 和后置条件 ψ 是完全正确的，当且仅当 M 对于前置条件 φ 和后置条件 ψ 是部分正确的且 M 对于前置条件 φ 是终止的。*

以公式为前置条件和后置条件的正确性问题: 相应地可以讨论以公式为前置条件和后置条件的正确性问题。 设 φ 和 ψ 为公式。我们可以把公式通过 I 解释成为谓词，则有类似的正确性问题的描述。 M 对于前置条件 φ 的终止性和 M 对于前置条件 φ 和后置条件 ψ 的部分正确性和完全正确性定义如下。

$$
\begin{array}{ll}
\text{终止性:} & \forall\sigma.(I(\varphi)(\sigma) \to \exists\sigma'.((beg,\sigma)\xrightarrow{*}(end,\sigma'))) \\
\text{部分正确性:} & \forall\sigma.(I(\varphi)(\sigma) \to \forall\sigma'.(((beg,\sigma)\xrightarrow{*}(end,\sigma')) \to I(\psi)(\sigma'))) \\
\text{完全正确性:} & \forall\sigma.(I(\varphi)(\sigma) \to \exists\sigma'.(((beg,\sigma)\xrightarrow{*}(end,\sigma')) \wedge I(\psi)(\sigma')))
\end{array}
$$

M 对于前置条件 φ 和后置条件 ψ 的部分正确性和完全正确性分别记作 $\models_I \{\varphi\}M\{\psi\}$ 和 $\models_I [\varphi]M[\psi]$ 。由完全正确性的定义知 M 对于前置条件 φ 的终止性为 $\models_I [\varphi]M[true]$ 。

§3.3.2 流程图模型与卫式迁移模型的等价

给定 (B,V) 上的流程图模型 M 和 B 的解释 $I=(D,I_0)$ 。
给定 $B'=(F\cup LB,P)$ 和 $V'=V\cup\{pc\}$ 。
给定 I_0' 为 I_0 的扩充，其扩充部分将 LB 中的符号解释到 D 中不同的元素。
给定 (B',V') 上的卫式迁移模型 $M'=(T',\Theta)$ 和 B' 的解释 $I'=(D,I_0')$ 。

Definition 3.11 *M 与 M' 是计算等价的，当且仅当对任意 M 的计算 $[(l_i,\sigma_i)]_{i\geq 0}$ 存在 M' 的计算 $[\sigma_i']_{i\geq 0}$ 满足 $\forall i\geq 0.((\forall\varphi.(\sigma_i\models_I\varphi\leftrightarrow\sigma_i'\models_{I'}\varphi))\wedge\sigma_i'\models_{I'} pc=l_i)$ ，反之亦然。*

Proposition 3.15 *设 M 与 $M'=(T',\Theta)$ 是计算等价的。设 φ,ψ 为公式。 M 对于前置条件 φ 和后置条件 ψ 是部分正确的当且仅当 $pc=end\to\psi$ 是 $M''=(T',\Theta\wedge\varphi)$ 的安全性质； M 对于前置条件 φ 是终止的当且仅当 $pc=end$ 是 $M''=(T',\Theta\wedge\varphi)$ 的必达性质。*

流程图模型的卫式迁移模型的构造: 给定 (B,V) 上的流程图模型 $M=(T)$ 和 B 的解释 $I=(D,I_0)$ 。
定义 $B'=(F\cup LB,P)$ 和 $V'=V\cup\{pc\}$ 。
定义 I_0' 为 I_0 的扩充，其扩充部分将 LB 中的符号解释到 D 中不同的元素。
设 N 为流程图模型指令的子集。
用 N 构造卫式迁移模型的迁移集合的算法 $g_1(N)$ ，定义如下。

若	N	$=$	$\{l:(x_1,...,x_k):=(e_1,...,e_k)\}\cup N_1$,
则	$g_1(N)$	$=$	$\{pc=l\longrightarrow(x_1,...,x_k,pc):=(e_1,...,e_k,l')\}\cup g_1(N_1)$;
若	N	$=$	$\{l:(if\ (b)\ goto\ l'\ else\ goto\ l''\ fi\}\cup N_1$,
则	$g_1(N)$	$=$	$\{pc=l\wedge b\longrightarrow(pc):=(l'),pc=l\wedge\neg e\longrightarrow(pc):=(l'')\}\cup g_1(N_1)$;
若	N	$=$	$\{\}$,
则	$g_1(N)$	$=$	$\{\}$ 。

定义 $M^*=(g_1(T),pc=beg)$ 。则 M^* 为卫式迁移模型。

Proposition 3.16 *$M=(T)$ 与 $M^*=(g_1(T),pc=beg)$ 是计算等价的。*

Corollary 3.4 *设 φ,ψ 为公式。 M 对于前置条件 φ 和后置条件 ψ 是部分正确的当且仅当 $pc=end\to\psi$ 是卫式迁移模型 $M'=(g_1(T),pc=beg\wedge\varphi)$ 的安全性质； M 对于前置条件 φ 是终止的当且仅当 $pc=end$ 是卫式迁移模型 M' 的必达性质。*

§3.4 结构化程序模型

结构化程序模型是一阶逻辑的扩充。 结构化程序模型的主要要素有赋值、顺序复合、条件语句和循环语句。 在一阶逻辑的基础上，我们增加以下集合的符号：

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

给定变量集合 V 。

Definition 3.12 一个 (B, V) 上的结构化程序模型是一个字符串，其集合 $\mathcal{L}_{\bigcirc}^{(B,V)}$ 定义如下。

S	$::=$	$\epsilon \mid T;\epsilon$
T	$::=$	$x := e \mid$
		$T;T \mid$
		$if\ b\ then\ T\ else\ T\ fi \mid$
		$while\ b\ do\ T\ od$

其中 $x \in V$ 为变量， $e \in T_{(B,V)}$ 为项， $b \in QFF_{(B,V)}$ 为公式。

为书写方便，在不引起歧义的情况下，模型结尾的 ϵ 可省略不写。

状态与状态空间: 结构化程序模型的系统状态有两个部分，即模型运行的剩余部分和变量状态。变量状态空间为 V 中变量取值的组合 $\Sigma = \{\sigma \mid \sigma(x) \in D, x \in V\}$ 。 系统的状态空间为模型和 V 中变量取值的组合 $\mathcal{L}_{\bigcirc}^{(B,V)} \times \Sigma$ 。

状态迁移关系: 系统状态的迁移关系 $\rightarrow$ 为 $(\mathcal{L}_{\bigcirc}^{(B,V)} \times \Sigma)^2$ 的子集。$(S_0, \sigma_0) \rightarrow (S_1, \sigma_1)$ 当且仅当以下一项成立。

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

结构化程序模型的迁移有如下特点。

Proposition 3.17 若 $(S, \sigma) \rightarrow (S', \sigma')$ 且 $(S, \sigma) \rightarrow (S'', \sigma'')$ 则 $(S', \sigma') = (S'', \sigma'')$ 。

计算: 结构化程序模型 M 的一个计算是状态集 $\mathcal{L}_{\bigcirc}^{(B,V)} \times \Sigma$ 上的一个无穷序列 $[(S_i, \sigma_i)]_{i \geq 0}$ 满足 $S_0 = M$ 且对任意 $i \geq 0$ 都有 $(S_i, \sigma_i) \rightarrow (S_{i+1}, \sigma_{i+1})$ 。

可达状态: M 在初始变量状态为 σ 时可到达的状态的集合为 $\{(M', \sigma') \mid (M, \sigma) \stackrel{*}{\rightarrow} (M', \sigma')\}$ 。M 在的可达状态集合为 $\{(M', \sigma') \mid (M, \sigma) \stackrel{*}{\rightarrow} (M', \sigma'), \sigma \in \Sigma\}$ 。

终止状态: M 在初始变量状态为 σ 时计算能够终止止，当且仅当存在 σ' 使得 $(M, \sigma) \stackrel{*}{\rightarrow} (\epsilon, \sigma')$ 。若存在 σ 使得 $(M, \sigma) \stackrel{*}{\rightarrow} (\epsilon, \sigma')$ ，则称 σ' 为 M 的一个终止状态。

计算结果的唯一性: 结构化程序模型的计算结果有唯一性。

Corollary 3.5 *给定结构化程序模型 S 和状态 σ 。若 $(S,\sigma)\xrightarrow{*}(\epsilon,\sigma')$ 且 $(S,\sigma)\xrightarrow{*}(\epsilon,\sigma'')$ 则 $\sigma'=\sigma''$ 。*

正确性性质: 设 $A\subseteq\mathcal{L}_{\bigcirc}^{(B,V)}\times\Sigma$ 。

- A 是 M 的安全性质，当且仅当 M 的所有计算都是 A 计算。
- A 是 M 的必达性质，当且仅当 M 的所有计算都能到达 A 状态。

§3.4.1 正确性问题

类似于流程图模型，结构化程序模型的基本正确性问题包括模型计算的终止性以及模型的终止状态和初始状态的关系。

终止性: 设 φ 是谓词。 M 对于前置条件 φ 的终止性定义如下。

$$\text{终止性：}\quad \forall\sigma.(\varphi(\sigma)\rightarrow\exists\sigma'.((M,\sigma)\xrightarrow{*}(\epsilon,\sigma')))$$

Proposition 3.18 *M 对于前置条件 $true$ 是终止的，当且仅当 $\{(\epsilon,\sigma)\mid\sigma\in\Sigma\}$ 是 M 的必达性质。*

部分正确性: 设 φ 和 ψ 是谓词。 M 对于前置条件 φ 和后置条件 ψ 的部分正确性定义如下。

$$\text{部分正确性：}\quad \forall\sigma.(\varphi(\sigma)\rightarrow\forall\sigma'.(((M,\sigma)\xrightarrow{*}(\epsilon,\sigma'))\rightarrow\psi(\sigma')))$$

Proposition 3.19 *M 对于前置条件 $true$ 和后置条件 ψ 是部分正确的，当且仅当 $\{(\epsilon,\sigma)|\psi(\sigma)\}\cup\{(T,\sigma)\mid T\neq\epsilon\}$ 是 M 的安全性质。*

完全正确性: 设 φ 和 ψ 是谓词。 M 对于前置条件 φ 和后置条件 ψ 的完全正确性定义如下。

$$\text{完全正确性：}\quad \forall\sigma.(\varphi(\sigma)\rightarrow\exists\sigma'.(((M,\sigma)\xrightarrow{*}(\epsilon,\sigma'))\wedge\psi(\sigma')))$$

Proposition 3.20 *M 对于前置条件 φ 和后置条件 ψ 是完全正确的，当且仅当 M 对于前置条件 φ 和后置条件 ψ 是部分正确的且 M 对于前置条件 φ 是终止的。*

以公式为前置条件和后置条件的正确性问题: 相应地可以定义以公式为前置条件和后置条件的正确性问题。 设 φ 和 ψ 为公式。 M 对于前置条件 φ 的终止性和 M 对于前置条件 φ 和后置条件 ψ 的部分正确性和完全正确性定义如下。

$$\begin{array}{ll}\text{终止性：} & \forall\sigma.(I(\varphi)(\sigma)\rightarrow\exists\sigma'.((M,\sigma)\xrightarrow{*}(\epsilon,\sigma')))\\ \text{部分正确性：} & \forall\sigma.(I(\varphi)(\sigma)\rightarrow\forall\sigma'.(((M,\sigma)\xrightarrow{*}(\epsilon,\sigma'))\rightarrow I(\psi)(\sigma')))\\ \text{完全正确性：} & \forall\sigma.(I(\varphi)(\sigma)\rightarrow\exists\sigma'.(((M,\sigma)\xrightarrow{*}(\epsilon,\sigma'))\wedge I(\psi)(\sigma')))\end{array}$$

M 对于前置条件 φ 和后置条件 ψ 的部分正确性和完全正确性分别记作 $\models_I\{\varphi\}M\{\psi\}$ 和 $\models_I[\varphi]M[\psi]$ 。由完全正确性可得出 M 对于前置条件 φ 的终止性为 $\models_I[\varphi]M[true]$ 。

§3.4.2 结构化程序模型与流程图模型的等价

给定 (B,V) 上的结构化程序模型 M 和流程图模型 M' 。

Definition 3.13 结构化程序模型 M 与流程图模型 M' 是计算等价的，当且仅当存在一一对应关系 $\zeta : LB \to \mathcal{L}_{\bigcirc}^{(B,V)}$ 满足 $\zeta(beg) = M$ 且 $\zeta(end) = \epsilon$ 使得 对任意 M 的计算 $[(S_i, \sigma_i)]_{i\geq 0}$ 存在 M' 的计算 $[(\zeta^{-1}(S_i), \sigma_i)]_{i\geq 0}$ ，反之亦然。

Proposition 3.21 设结构化程序模型 M 与流程图模型 M' 是计算等价的。设 φ, ψ 为公式。M 对于前置条件 φ 和后置条件 ψ 是部分正确的当且仅当 M' 对于 φ 和 ψ 是部分正确的， M 对于前置条件 φ 是终止的当且仅当 M' 对于 φ 是终止的。

结构化程序模型的流程图模型的构造: 给定 (B,V) 上的结构化程序模型 M 和 B 的解释 $I = (D, I_0)$ 。

对 M 中的每个语句 T 的入口标记一个唯一标号 $lb(T)$ 满足 $lb(M) = beg$ 且 $lb(\epsilon) = end$ 且同一语句若出现在不同地方则赋予不同标号。 记打上标号的 M 为 M_{lb} 。

我们用带标号的结构化程序模型 M_{lb} 构造流程图模型。构造算法 $g_2(S)$ 其输入 S 为 M_{lb} 的片段，定义如下。

S	$g_2(S)$
$end : \epsilon$	$\{\}$
l : x:=t; S_1	$\{l : (x) := (t) \text{ goto } lb(S_1)\} \cup g_2(S_1)$
l : if b then T_1 else T_2 fi; S_1	$\{l : \text{if } (b) \text{ goto } lb(T_1) \text{ else goto } lb(T_2) \text{ fi}\} \cup g_2(T_1; S_1) \cup g_2(T_2; S_1)$
l : while b do T_1 od; S_1	$\{l : \text{if } (b) \text{ goto } lb(T_1) \text{ else goto } lb(S_1) \text{ fi}\} \cup g_2(T_1; S_1)$

定义 $M^* = g_2(M_{lb})$ 。则 M^* 为流程图模型。

Proposition 3.22 M 与 $M^* = g_2(M_{lb})$ 是计算等价的。

Corollary 3.6 设 φ, ψ 为公式。 M 对于前置条件 φ 和后置条件 ψ 是部分正确的当且仅当 $M^* = g_2(M_{lb})$ 对于 φ 和 ψ 是部分正确的， M 对于前置条件 φ 是终止的当且仅当 M^* 对于 φ 是终止的。

§3.5 练习

1. 证明流程图模型的计算结果有唯一性，即对任意给定流程图模型 M 和任意给定状态 σ ，若 $(beg, \sigma) \xrightarrow{*} (end, \sigma')$ 且 $(beg, \sigma) \xrightarrow{*} (end, \sigma'')$ 则 $\sigma' = \sigma''$ 。
2. 证明结构化程序模型的计算结果有唯一性，即对任意给定结构化程序模型 S 和任意给定状态 σ ，若 $(S, \sigma) \xrightarrow{*} (\epsilon, \sigma')$ 且 $(S, \sigma) \xrightarrow{*} (\epsilon, \sigma'')$ 则 $\sigma' = \sigma''$ 。
3. 定义一种流程图模型与标号 Kripke 结构的合理的等价关系和一种满足所定义等价关系的由流程图模型构造标号 Kripke 模型的算法。

公平安全性质的推理: 我们有以下规则。

$$\frac{\begin{array}{l}\text{对所有 } i = 1, ..., k \text{ 都有 } \phi_i \text{ 是 } (M, \psi_i) \text{ 的安全性质} \\ \bigvee_{i=1}^{k} \psi_i \text{ 是 } (M, \psi) \text{ 的必达性质} \\ \varphi \vee \psi \text{ 是 } M' = \langle S, R, I, L\rangle \text{ 的安全性质}\end{array}}{\varphi \text{ 是 } M \text{ 的公平安全性质}}$$

练习 3.1 证明结构化程序模型的计算结果有唯一性，即对任意给定结构化程序模型 S ， 对任意 σ ，若 $(S,\sigma)\xrightarrow{*}(\epsilon,\sigma')$ 且 $(S,\sigma)\xrightarrow{*}(\epsilon,\sigma'')$ 则 $\sigma'=\sigma''$ 。

练习 3.2 定义一种流程图模型与标号 *Kripke* 结构的合理的等价关系和一种满足以上等价关系的由流程图模型构造标号 *Kripke* 结构的算法。

§4 标号迁移模型

本章介绍以标号的迁移为特点的模型。 这类模型除了与 Kripke 模型类似的基本元素外，迁移有自己的标号，便于描述一个计算是通过什么样的迁移或什么类型的迁移进行的。

§4.1 标号迁移系统

Definition 4.1 一个标号迁移系统是一个四元组 $\langle\Sigma,S,\Delta,I\rangle$ 其中 Σ 为标号的有穷集合， S 为有穷状态集合， $\Delta\subseteq S\times\Sigma\times S$ 为标号迁移关系， $I\subseteq S$ 为初始状态集。

以下假定标号迁移系统 $M=\langle\Sigma,S,\Delta,I\rangle$ 为给定。

字符串上的运行: 用 $s\xrightarrow{a}s'$ 表示存在从 s 到 s' 的 a 迁移，即 $(s,a,s')\in\Delta$ 。 给定一个 Σ 上的字符串 $w=[a_i]_{i\geq 1}$ 。状态集 S 上的无穷序列 $\pi=[s_i]_{i\geq 0}$ 是 w 上的一个运行，当且仅当对所有 $i\geq 0$ 有 $s_i\xrightarrow{a_{i+1}}s_{i+1}$ 。

运行: 状态集 S 上的无穷序列 π 是 M 的一个运行，当且仅当存在 Σ 上的字符串 w 使得 π 是 w 上的一个运行。

运行上的字符串: 给定一个 Σ 上的字符串 $w=[a_i]_{i\geq 1}$ 。 w 是运行 $\pi=[s_i]_{i\geq 0}$ 上的字符串，当且仅当对所有 $i\geq 0$ 有 $s_i\xrightarrow{a_{i+1}}s_{i+1}$ 。

确定型与非确定型系统: 我们可以将标号迁移系统分为确定型系统和非确定型系统。确定型系统的初始状态只有一个且其迁移关系满足以下限制：

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

Proposition 4.1 给定一个字符串 w 。对确定型系统，若 π 和 π' 是 w 上的运行，则 $\pi=\pi'$ 。

双标号迁移系统: 与标号 Kripke 结构类似，我们可以在标号迁移系统的状态上添加一些信息，构造双标号迁移系统。设 AP 为给定命题集合。

Definition 4.2 一个 AP 上的双标号迁移系统是一个五元组 $\langle\Sigma,S,\Delta,I,L\rangle$ 其中 $\langle\Sigma,S,\Delta,I\rangle$ 为标号迁移系统， $L:S\rightarrow 2^{AP}$ 为状态标号函数。

§4.2 无穷字符串上的自动机

对于标号迁移系统而言，运行就是满足迁移关系的起点为初始状态的无穷序列。为了增加表达能力，更精确地描述一个系统，我们可以对运行做限制以排除一些部分这样的无穷状态序列。

Definition 4.3 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， $F\subseteq S$ 为状态集。

可接受运行： Büchi 自动机 $\mathcal{B} = \langle \Sigma, S, \Delta, I, F \rangle$ 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 $\pi = [s_i]_{i \geq 0}$ 是可接受的当且仅当

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

可接受字符串与自动机语言：一个字符串 $w = [a_i]_{i \geq 1}$ 是可接受的当且仅当存在 w 上的可接受运行。 $\mathcal{B}$ 的可接受字符串的集合为 $\mathcal{B}$ 的语言，记作 $\mathcal{L}(\mathcal{B})$ 。

语言非空问题：自动机 B 的语言为空，即 $\mathcal{L}(B) = \emptyset$ ，记作 $B \equiv \emptyset$ 。

Proposition 4.2 *给定 Büchi 自动机 $B = \langle \Sigma, S, \Delta, I, F \rangle$ 。令 $R = \{(s, s') \mid (s, a, s') \in \Delta\}$ 。$B \equiv \emptyset$ 当且仅当公平 Kripke 模型 $K = \langle S, R, I, \{F\} \rangle$ 为空当且仅当 $EmpChecking(K)$ 为 $true$ 。*

自动机的运算：自动机的运算包括交、并和补。 A' 是 A 和 B 的交自动机，记作 $A =' A \cap B$ ，当前仅当 $\mathcal{L}(A') = \mathcal{L}(A) \cap \mathcal{L}(B)$ 。 A' 是 A 和 B 的并自动机，记作 $A' = A \cup B$ ，当前仅当 $\mathcal{L}(A') = \mathcal{L}(A) \cup \mathcal{L}(B)$ 。 A' 是 A 的补自动机，记作 $A' = \neg A$ ，当前仅当 $\mathcal{L}(A') = \Sigma^\omega \setminus \mathcal{L}(A)$ 。

Definition 4.4 *Büchi 自动机是交封闭的，当且仅当对任意 Büchi 自动机 B_1 和 B_2 ，存在一个 Büchi 自动机 B 使得 $\mathcal{L}(B) = \mathcal{L}(B_1) \cap \mathcal{L}(B_2)$ 。*

Proposition 4.3 *Büchi 自动机是交封闭的。*

以下是一种构造交自动机的算法。
给定两个 Büchi 自动机 $B_1 = \langle \Sigma, S_1, \Delta_1, I_1, F_1 \rangle$, $B_2 = \langle \Sigma, S_2, \Delta_2, I_2, F_2 \rangle$ 。
定义 $S = S_1 \times S_2 \times \{0, 1, 2\}$ 且 Δ 为以下集合的并集。

$\{((s_1, s_2, i), a, ((s_1', s_2', i)) \mid (s_1, a, s_1') \in \Delta_1, (s_2, a, s_2') \in \Delta_2, i \in \{0, 1\}\}$
$\{((s_1, s_2, i), a, ((s_1', s_2', i+1)) \mid (s_1, a, s_1') \in \Delta_1, (s_2, a, s_2') \in \Delta_2, i \in \{0, 1\}, s_{i+1} \in F_{i+1}\}$
$\{((s_1, s_2, 2), a, ((s_1', s_2', 0)) \mid (s_1, a, s_1') \in \Delta_1, (s_2, a, s_2') \in \Delta_2\}$

定义 $B_1 \cap B_2 = \langle \Sigma, S, \Delta, I_1 \times I_2 \times \{0\}, S_1 \times S_2 \times \{2\} \rangle$ 。 则 $\mathcal{L}(B_1 \cap B_2) = \mathcal{L}(B_1) \cap \mathcal{L}(B_2)$ 。

Definition 4.5 *Büchi 自动机是并封闭的，当且仅当对任意 Büchi 自动机 B_1 和 B_2 ，存在一个 Büchi 自动机 B 使得 $\mathcal{L}(B) = \mathcal{L}(B_1) \cup \mathcal{L}(B_2)$ 。*

Proposition 4.4 *Büchi 自动机是并封闭的。*

以下是一种构造并自动机的算法。
给定两个 Büchi 自动机 $B_1 = \langle \Sigma, S_1, \Delta_1, I_1, F_1 \rangle$, $B_2 = \langle \Sigma, S_2, \Delta_2, I_2, F_2 \rangle$ 且 $S_1 \cap S_2 = \emptyset$ 。
定义 $B_1 \cup \mathcal{B}_2 = \langle \Sigma, S_1 \cup S_2, \Delta_1 \cup \Delta_2, I_1 \cup I_2, F_1 \cup F_2 \rangle$ 。 则 $\mathcal{L}(B_1 \cup B_2) = \mathcal{L}(B_1) \cup \mathcal{L}(B_2)$ 。

Definition 4.6 *Büchi 自动机是补封闭的，当且仅当对任意 Büchi 自动机 B ，存在一个 Büchi 自动机 B' 使得 $\mathcal{L}(B') = \Sigma^\omega \setminus \mathcal{L}(B)$ 。*

Proposition 4.5 *Büchi 自动机是补封闭的。*

以下是一种构造补自动机的算法。
给定 Büchi 自动机 $B = \langle \Sigma, S, \Delta, I, F \rangle$ 。我们将 $\alpha \in \Sigma^*$ 根据 $p, q \in S$ ，按以下条件进行分类： 可连接 p, q 且可经过一个 F 状态的； 可连接 p, q 且不能经过 F 状态的； 不能连接 p, q 的。记 n 为状态的数目。类型的总数不超过 3^{n^2} 。记这些类型为 $t_1, ..., t_m$ 。
由拉姆齐 定理知，任意无穷串可写为 $t_i(t_j)^\omega$ 的形式。

因而 $\Sigma^\omega = \bigcup\{t_i(t_j)^\omega \mid 1 \le i,j \le m\}$ 。
由分类的构造知， $t_i(t_j)^\omega \subseteq \mathcal{L}(B)$ 或 $t_i(t_j)^\omega \subseteq \Sigma^\omega \setminus \mathcal{L}(B)$ 。
因而 $\Sigma^\omega \setminus L(B) = \bigcup\{t_i(t_j)^\omega \mid 1 \le i,j \le m, t_i(t_j)^\omega \not\subseteq \mathcal{L}(B)\}$ 。
由每类 t_i 是一个有穷字上的自动机的语言知， $t_i(t_j)^\omega$ 是一个 Büchi 自动机的语言。
定义 $A_{i,j}$ 使得其语言为 $t_i(t_j)^\omega$ 。
定义 $B' = \bigcup\{A_{i,j} \mid 1 \le i,j \le m, A_{i,j} \cap B \equiv \emptyset\}$ 。 则 B' 是 Büchi 自动机 B 的补自动机。

语言包含问题: 自动机 A 的语言是否包含于自动机 B 的语言，即 $\mathcal{L}(A) \subseteq \mathcal{L}(B)$ 是否成立， 可通过自动机的交和补运算转换成语言非空问题。 $\mathcal{L}(A) \subseteq \mathcal{L}(B)$ 当且仅当 $\mathcal{L}(A) \cap (\Sigma^\omega \setminus \mathcal{L}(B)) = \emptyset$ 当且仅当 $A \cap \neg B \equiv \emptyset$ 。

§4.2.1 自动机的可接受运行条件的不同设置

我们可以根据不同的需要设置不同的可接受运行的条件。这些条件的主要类型有以下几种。

泛 Büchi 自动机:

Definition 4.7 *一个泛 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， $F \subseteq 2^S$ 为状态集的集合。*

泛 Büchi 自动机 $\mathcal{B} = \langle \Sigma, S, \Delta, I, F\rangle$ 上的运行就是标号迁移系统 $\langle \Sigma, S, \Delta, I\rangle$ 的运行。 $\mathcal{B}$ 上的运行 $\pi = [s_i]_{i\ge 0}$ 是可接受的当且仅当 $\forall f \in F.(inf(\pi) \cap f \neq \emptyset)$.

Muller 自动机:

Definition 4.8 *一个 Muller 自动机是一个五元组 $\langle \Sigma, S, \Delta, I, F\rangle$ 其中 $\langle \Sigma, S, \Delta, I\rangle$ 是一个标号迁移系统， $F \subseteq 2^S$ 为状态集的集合。*

Muller 自动机 $\mathcal{B} = \langle \Sigma, S, \Delta, I, F\rangle$ 上的运行就是标号迁移系统 $\langle \Sigma, S, \Delta, I\rangle$ 的运行。 $\mathcal{B}$ 上的运行 $\pi = [s_i]_{i\ge 0}$ 是可接受的当且仅当 $inf(\pi) \in F$.

Streett 自动机:

Definition 4.9 *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， $F \subseteq 2^S \times 2^S$ 为状态集合对的集合。*

Streett 自动机 $\mathcal{B} = \langle \Sigma, S, \Delta, I, F\rangle$ 上的运行就是标号迁移系统 $\langle \Sigma, S, \Delta, I\rangle$ 的运行。 $\mathcal{B}$ 上的运行 $\pi = [s_i]_{i\ge 0}$ 是可接受的当且仅当对所有 $(f,g) \in F$ ， $inf(\pi) \cap f \neq \emptyset \to inf(\pi) \cap g \neq \emptyset$.

Rabin 自动机:

Definition 4.10 *一个 Rabin 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， $F \subseteq 2^S \times 2^S$ 为状态集合对的集合。*

Rabin 自动机 $\mathcal{B} = \langle \Sigma, S, \Delta, I, F\rangle$ 上的运行就是标号迁移系统 $\langle \Sigma, S, \Delta, I\rangle$ 的运行。 $\mathcal{B}$ 上的运行 $\pi = [s_i]_{i\ge 0}$ 是可接受的当且仅当存在 $(f,g) \in F$ ， $inf(\pi) \cap f \neq \emptyset \wedge inf(\pi) \cap g = \emptyset$.

Parity 自动机:

Definition 4.11 *一个 Parity 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, $F : S \to N$(其中 N 为自然数集合) 为优先级函数。*

Parity 自动机 $\mathcal{B} = \langle \Sigma, S, \Delta, I, F\rangle$ 上的运行就是标号迁移系统 $\langle \Sigma, S, \Delta, I\rangle$ 的运行。 $\mathcal{B}$ 上的运行 $\pi = [s_i]_{i\geq 0}$ 是可接受的当且仅当以下集合的最小元为偶数: $\{F(s) \mid s \in inf(\pi)\}$.

自动机的等价: 一个自动机等价于另一个自动机当且仅当其所定义的语言是相同的。两类自动机的表达能力相同, 当且仅当任意一类自动机的一个实例都有一个等价的另一类自动机的一个实例。

Proposition 4.6 *Büchi 自动机、泛 Büchi 自动机、 Muller 自动机、 Streett 自动机、 Rabin 自动机和 Parity 自动机的表达能力是相同的。*

容易验证 Büchi 自动机的条件是以上自动机都可以表达的, 而以上自动机的条件都是 Muller 自动机可以表达的。上述命题得证当且仅当对任意 Muller 自动机都有与之等价的 Büchi 自动机。

给定 Muller 自动机 $\mathcal{B} = \langle \Sigma, S, \Delta, I, \{F_0, ..., F_n\}\rangle$ 。

定义 $S' = S \cup \bigcup_{i=0}^{n}\{i\} \times F_i \times 2^{F_i}$ 且 Δ' 为以下集合。

$\{(s, a, (i, s', \emptyset)) \mid (s, a, s') \in \Delta, s' \in F_i\}\cup$
$\{((i, s, R), a, (i, s', R \cup \{s\})) \mid (s, a, s') \in \Delta, R \neq F_i, s, s' \in F_i\}\cup$
$\{((i, s, F_i), a, (i, s', \emptyset)) \mid (s, a, s') \in \Delta, s, s' \in F_i\} \cup \Delta$

定义 Büchi 自动机 $B' = \langle \Sigma, S', \Delta', I, \bigcup_{i=0}^{n}\{i\} \times F_i \times \{F_i\}\}\rangle$ 。 则 $\mathcal{L}(B') = \mathcal{L}(B)$ 。

§4.2.2 确定型自动机与非确定型自动机

一类自动机称为确定型的该类自动机当且仅当其标号迁移系统是确定型。不受确定型条件限制的自动机亦称为非确定型自动机。

Proposition 4.7 *确定型 Büchi 自动机不是补封闭的。*

Proposition 4.8 *确定型 Muller 自动机、确定型 Streett 自动机、确定型 Rabin 自动机和确定型 Parity 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。*

§4.2.3 自动机与 Kripke 模型

Definition 4.12 *给定泛 Büchi 自动机 $B = \langle \Sigma, S, \Delta, I, F\rangle$ 和公平标号 Kripke 模型 K 。B 与 K 是等价的, 记作 $B \equiv K$, 当且仅当 $\mathcal{L}(B) = \mathcal{L}(K)$, 或对任意 $[w_i]_{i\geq 0} \in \mathcal{L}(B)$ 存在 $[w'_i]_{i\geq 0} \in \mathcal{L}(K)$ 使得对所有 $i \geq 0$ 都有 $\{w_i\} = w'_i \cap \Sigma$ 且对任意 $[w'_i]_{i\geq 0} \in \mathcal{L}(K)$ 存在 $[w_i]_{i\geq 0} \in \mathcal{L}(B)$ 使得对所有 $i \geq 0$ 都有 $\{w_i\} = w'_i \cap \Sigma$ 。*

对于自动机我们可以构造与之等价的公平标号 Kripke 模型。反之亦然。

公平标号 Kripke 模型到泛 Büchi 自动机的构造: 给定 AP 上的公平标号 Kripke 模型 $K = \langle S, R, I, L, F\rangle$ 。我们构造与之等价的泛 Büchi 自动机如下。

定义 $\Sigma = 2^{AP}$ 。

定义 $\Delta = \{(s, a, s') \mid (s, s') \in R, a = L(s)\}$ 。

定义泛 Büchi 自动机 $B_M = \langle \Sigma, S, \Delta, I, \{[[\varphi]] \mid \varphi \in F\}\rangle$ 。则 $\mathcal{L}(B) = \mathcal{L}(K)$ 。

泛 Büchi 自动机到公平标号 Kripke 模型的构造: 给定泛 Büchi 自动机 $B = \langle \Sigma, S, \Delta, I, F \rangle$ 且 $F = \{f_1, ..., f_n\}$ 。我们构造与之等价的公平标号 Kripke 模型如下。

定义 $AP = \Sigma \cup \{p_1, ..., p_n\}$ 。

定义 $S' = \Delta$ 。

定义 $R' = \{((s, a, s'), (s', b, s'')) \mid (s, a, s'), (s', b, s'') \in S'\}$ 。

定义 $I' = \{(s, a, s') \mid (s, a, s') \in S', s \in I\}$

定义 $L((s, a, s')) = \{a\} \cup \{p_i \mid s \in f_i, i = 1, ..., n\}$ 。

定义 $F' = \{p_1, ..., p_n\}$ 。

定义 AP 上的公平标号 Kripke 模型 $K = \langle S', R', I', L, F' \rangle$ 。

则对任意 $[w_i]_{i \geq 0} \in \mathcal{L}(B)$ 存在 $[w'_i]_{i \geq 0} \in \mathcal{L}(K)$ 使得对所有 $i \geq 0$ 都有 $\{w_i\} = w'_i \cap \Sigma$ 且对任意 $[w'_i]_{i \geq 0} \in \mathcal{L}(K)$ 存在 $[w_i]_{i \geq 0} \in \mathcal{L}(B)$ 使得对所有 $i \geq 0$ 都有 $\{w_i\} = w'_i \cap \Sigma$ 。

§4.3 时间迁移系统与时间自动机

对一些系统，我们除了关系系统的控制状态外，可能还关心在一个状态持续的时间长短或两个状态之间的时间变化。为了能够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。

时钟公式: 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 (有理数) 的集合。时钟公式的集合 $\Phi(X)$ 由以下语法给出。

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

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。 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' 。

Definition 4.13 一个时间迁移系统是一个五元组 $\langle \Sigma, S, X, \Delta, I \rangle$ 其中 Σ 为字母表， S 为有穷状态集合， X 为时钟变量集合， $\Delta \subseteq S \times \Sigma \times 2^X \times \Phi(X) \times S$ 为迁移关系， $I \in S$ 为模型初始状态的集合。

给定时间迁移系统 $M = \langle \Sigma, S, X, \Delta, I \rangle$ 。

状态空间: 模型状态由两部分组成，一部分是状态 $s \in S$ ，另一部分为时间变量的状态，称为时钟赋值。记 $V = X \to R$ 为时钟赋值的集合。 M 的状态空间为 $S \times V$ 。

时间迁移与动作迁移: 设 t 为时间常量。定义 $(s, v) \xrightarrow{t} (s, v')$ 当且仅当 $v' = v + t$ 。设 $\sigma \in \Sigma$ 。定义 $(s, v) \xrightarrow{\sigma} (s', v')$ 当且仅当存在 $(s, \sigma, \lambda, \varphi, s') \in \Delta$ 使得 $v \models \varphi$ 且 $v' = [\lambda \to 0]v_i$ 。

迁移关系: 定义 $(s, v) \overset{(\sigma,t)}{\to} (s', v')$ 当且仅当存在 v'' 使得 $(s, v) \xrightarrow{t} (s, v'')$ 且 $(s, v'') \xrightarrow{\sigma} (s, v')$ 。 M 的迁移关系 $\to$ 是 $(S \times V)^2$ 的子集。 $(s, v) \to (s', v')$ 当且仅当存在 (σ, t) 使得 $(s, v) \overset{(\sigma,t)}{\to} (s', v')$ 。

时间字符串上的运行: 一个时间字符串为 $\Sigma \times R$ 上的无穷序列。为方便起见，有时也写成 $(\sigma, \tau) = ([\sigma_i]_{i \geq 1}, [\tau_i]_{i \geq 1}) \in \Sigma^\omega \times R^\omega$ 其中其中 τ 满足对所有 i, $\tau_{i+1} > \tau_i$ 且对任意 $t \in R$ 存在 i, $\tau_i > t$ 。为方便建摸，有些模型将 $\tau_{i+1} > \tau_i$ 的要求弱化为 $\tau_{i+1} \geq \tau_i$ 。

定义 $\tau_0 = 0$ 。时间迁移系统 $\langle \Sigma, S, X, \Delta, I \rangle$ 在时间字符串 $(\sigma, \tau) = [(\sigma_i, \tau_i)]_{i \geq 1} \in (\Sigma \times R)^\omega$ 上的一个运行是状态集 $S \times V$ 上的一个无穷序列 $[(s_i, v_i)]_{i \geq 0}$ 满足

$s_0 \in I$,
$v_0 = [X \to 0]v_0$,
对所有 $i \geq 0$ ， $(s_i, v_i) \overset{(\sigma_{i+1}, \tau_{i+1}-\tau_i)}{\to} (s_{i+1}, v_{i+1})$ 。

运行: $S \times V$ 上的一个无穷序列 r 是时间迁移系统 $\langle \Sigma, S, X, \Delta, I \rangle$ 的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。

可达状态: 给定时间迁移系统 $M = \langle \Sigma, S, X, \Delta, I \rangle$ 。一个状态 (s, v) 可达当且仅当存在 $s_0 \in I$ 使得 $(s_0, [X \to 0]v) \overset{*}{\to} (s, v)$ 。M 的可达状态集合为 $\{(s, v) | (s_0, v_0) \overset{*}{\to} (s, v), s_0 \in I, v_0 = [X \to 0]v_0\}$ 。

可达性问题和安全问题: 可达性问题是指给定一个模型和一组条件，判断模型是否有可达状态满足给定的条件。安全问题是指给定一个模型和一组条件，判断模型的所有可达状态是否都满足给定的条件。尽管时间迁移系统的可达状态集合通常是无穷的，时间迁移系统的状态可达性问题是可判定的。

Definition 4.14 *一个时间 Büchi 自动机是一个六元组 $\langle \Sigma, S, X, \Delta, I, F \rangle$ 其中 $\langle \Sigma, S, X, \Delta, I \rangle$ 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 $\mathcal{B} = \langle \Sigma, S, X, \Delta, I, F \rangle$ 。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。

可接受时间字符串与自动机语言: 一个时间字符串 (σ, τ) 是可接受的当且仅当存在 (σ, τ) 的可接受运行。 $\mathcal{B}$ 的可接受时间字符串的集合为 $\mathcal{B}$ 的语言，记作 $\mathcal{L}(\mathcal{B})$ 。 时间自动机是并和交封闭的，但不是补封闭的。

§4.4 混成迁移系统

混成迁移系统是时间迁移系统的一种扩充。设 $X = \{x_1, ..., x_n\}$ 为实数变量的集合。每个实数变量都是时间的函数。给定 $x \in X$ 。记 $\dot{x}$ 为 x 的导函数。记 $\dot{X} = \{\dot{x_1}, ..., \dot{x_n}\}$ 。记 $\Phi(X)$ 为 X 上谓词的集合。记 $\uplus(X) : X \hookrightarrow (R^n \to R)$ 为偏函数的集合。

Definition 4.15 *一个混成迁移系统 (也称为混成自动机) 是一个六元组 $\langle \Sigma, S, X, \Delta, I, flow \rangle$ 其中 Σ 为字母表， S 为有穷状态集合， X 为实数变量集合， $\Delta \subseteq S \times \Sigma \times \uplus(X) \times \Phi(X) \times S$ 为迁移关系， $I \in S \times (2^R)^n$ 为模型初始状态的集合， $flow : S \to \Phi(X \cup \dot{X})$ 为每个状态标上一个变量变化的约束条件。*

给定混成迁移系统 $\langle \Sigma, S, X, \Delta, I, flow \rangle$ 。

状态空间: 模型状态由两部分组成，一部分是状态 $s \in S$ ，另一部分为变量的状态，称为赋值。记 $V = X \to R$ 为赋值的集合。 M 的状态空间为 $S \times V$ 。

时间迁移与动作迁移: 设 $\delta \in R_{\geq 0}$ 为时间常量。定义 $(s,v) \xrightarrow{\delta} (s,v')$ 当且仅当存在一个可导函数 $f:[0,\delta] \to R^n$ 和其一阶导数 $\dot{f}:(0,\delta) \to R^n$ 满足 $f(0)=v$, $f(\delta)=v'$ 且

$$\forall \zeta \in (0,\delta).flow(s)(f(\zeta)/X, \dot{f}(\zeta)/\dot{X}) = true.$$

设 $\sigma \in \Sigma$ 。定义 $(s,v) \xrightarrow{\sigma} (s',v')$ 当且仅当存在 $(s,\sigma,\lambda,\varphi,s') \in \Delta$ 且 $\{z_1,...,z_k\} = dom(\lambda)$ 使得 $v \models \varphi$ 且 $v' = v[z_1/\lambda(z_1)(v)] \cdots [z_k/\lambda(z_k)(v)]$ 。

迁移关系: 定义 $(s,v) \overset{(\sigma,t)}{\to} (s',v')$ 当且仅当存在 v'' 使得 $(s,v) \xrightarrow{t} (s,v'')$ 且 $(s,v'') \xrightarrow{\sigma} (s,v')$ 。 M 的迁移关系 $\to$ 是 $(S \times V)^2$ 的子集。 $(s,v) \to (s',v')$ 当且仅当存在 (σ,t) 使得 $(s,v) \overset{(\sigma,t)}{\to} (s',v')$ 。

时间字符串上的运行: 一个时间字符串为 $\Sigma \times R$ 上的无穷序列。混成迁移系统 $\langle \Sigma, S, X, \Delta, I, flow \rangle$ 在时间字符串 $(\sigma,\tau) = [(\sigma_i,\tau_i)]_{i\geq 1} \in (\Sigma \times R)^\omega$ 上的一个运行是状态集 $S \times V$ 上的一个无穷序列 $[(s_i,v_i)]_{i\geq 0}$ 满足 $(s_0,v_0) \in I$ 且对任意 $i \geq 0$ ， $(s_i,v_i) \overset{(\sigma_{i+1},\tau_{i+1}-\tau_i)}{\longrightarrow} (s_{i+1},v_{i+1})$ 。

运行: $S \times V$ 上的一个无穷序列 r 是混成迁移系统 $\langle \Sigma, S, X, \Delta, I, flow \rangle$ 一个运行当且仅当存在一个时间字符串 (σ,τ) 使得 r 是混成迁移系统在 (σ,τ) 上的一个运行。

可达状态: 给定混成迁移系统 $M = \langle \Sigma, S, X, \Delta, I, flow \rangle$ 。一个状态 (s,v) 可达当且仅当存在 $(s_0,v_0) \in I$ 使得 $(s_0,v_0) \xrightarrow{*} (s,v)$ 。 M 的可达状态集合为 $\{(s,v) \mid (s_0,v_0) \xrightarrow{*} (s,v), (s_0,v_0) \in I\}$ 。

可达性问题和安全问题: 可达性问题是指给定一个模型和一组条件，判断模型是否有可达状态满足给定的条件。安全问题是指给定一个模型和一组条件，判断模型的所有可达状态是否都满足给定的条件。

§4.5 Petri 网模型

Petri 网模型的 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 4.16 *一个 Petri 网模型是一个四元组 $\langle P,T,F,m_0 \rangle$ 其中 P 为库所的有穷集合， T 为迁移的有穷集合， $F \subseteq (P \times T) \cup (T \times P)$ 为边的集合， $m_0 : P \to N$ 为库所集合到自然数集合的函数，表示初始状态。*

给定 $M = \langle P,T,F,m_0 \rangle$ 。
对 $t \in T$ ，定义 ${}^\circ p(t) = \{p \in P \mid (p,t) \in F\}$ 和 $p^\circ(t) = \{p \in P \mid (t,p) \in F\}$ 。
${}^\circ p(t)$ 和 $p^\circ(t)$ 分别为 t 的输入库所和输出库所的集合。

状态空间: 模型状态是所有库所的状态的组合，表示为对库所进行赋值的一个函数。模型的状态空间为 $V = P \to N$ 。

可执行迁移: 迁移 $t \in T$ 在状态 m 是可执行的，当且仅当 $\forall p \in {}^\circ p(t).(m(p) \geq 1)$ 。

迁移关系 (交错语义) : 定义 $\alpha_0(p,t) = 1$ 当且仅当 $p \in {}^\circ p(t)$ 。定义 $\alpha_1(p,t) = 1$ 当且仅当 $p \in p^\circ(t)$ 。 定义 $m \xrightarrow{t} m'$ 当且仅当迁移 $t \in T$ 在状态 m 可执行且

$$\forall p \in P.(m'(p) = m(p) - \alpha_0(p,t) + \alpha_1(p,t)).$$

M 的迁移关系 $\to$ 是 V^2 的子集。 $m \to m'$ 当且仅当存在可执行迁移 t 使得 $m \xrightarrow{t} m'$ 或不存在可执行迁移且 $m = m'$ 。

运行: 状态空间 V 上的一个无穷序列 $[m_i]_{i\geq 0}$ 是 M 的一个运行, 当且仅当对所有 i 有 $m_i \to m_{i+1}$ 。

可达状态: 给定 $M = \langle P, T, F, m_0\rangle$ 。一个状态 m 是可达的当且仅当 $m_0 \xrightarrow{*} m$ 。 M 的可达状态集合为 $\{m \mid m_0 \xrightarrow{*} m\}$ 。

可达性问题: 状态的可达性问题是指给定一个模型和一个状态，判断这个状态是否可达。 尽管 Petri 网模型的可达状态集合通常是无穷的， Petri 网模型的状态可达性问题是可判定的。

活性性质: Petri 网模型的活性性质是指迁移是否可执行或者以至于是否能执行无限多次。根据执行次数的区别可定义 L_1 活性 (迁移有可能执行) 和 L_2 活性 (迁移有可能执行任意多次) 等性质。

有穷状态模型: Petri 网模型 $\langle P, T, W, m_0\rangle$ 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。

加权 Petri 网模型: 为方便建摸，我们可以对 Petri 网模型的边增加权重，建立加权 Petri 网模型。

Definition 4.17 *一个加权 Petri 网模型是一个四元组 $\langle P, T, W, m_0\rangle$ 其中 P ， T 和 m_0 同于以上 Petri 网模型， $W : (P \times T) \cup (T \times P) \to N$ 为具有权重的边的集合。*

给定 $M = \langle P, T, W, m_0\rangle$ 。 M 的状态空间同于以上 Petri 网模型。迁移 $t \in T$ 在状态 m 是可执行的，当且仅当 $\forall p \in {}^\circ p(t).(m(p) \geq W(p, t))$ 。 $m \xrightarrow{t} m'$ 当且仅当迁移 $t \in T$ 在状态 m 可执行且

$$\forall p \in P.(m'(p) = m(p) - W(p, t) + W(t, p)).$$

M 的迁移关系 $\to$ ，运行集合和可达状态集合的定义类似于普通 Petri 网模型。加权 Petri 网模型并不增加表达能力。对加权 Petri 网模型我们可以构造等效的 (可以映射到同样状态序列的) 普通 Petri 网模型。

Proposition 4.9 *每一个加权 Petri 网模型都可以转换成一个等效的普通 Petri 网模型。*

容量限制的 Petri 网模型: 我们可以对 Petri 网模型的库所进行容量限制, 建立容量限制的 Petri 网模型。

Definition 4.18 *一个容量限制的 Petri 网模型是一个五元组 $\langle P, T, F, K, m_0\rangle$ 其中 $\langle P, T, F, m_0\rangle$ 为普通 Petri 网模型， $K : P \to N$ 为容量限制函数。*

给定 $M = \langle P, T, F, K, m_0\rangle$ 。 M 的状态空间同于以上 Petri 网模型。迁移 $t \in T$ 在状态 m 是可执行的，当且仅当 $\forall p \in {}^\circ p(t).(m(p) \geq 1)$ 且 $\forall p \in p^\circ(t).(m(p) - \alpha_0(p, t) + \alpha_1(p, t) \leq K(p))$. $m \xrightarrow{t} m'$ 当且仅当迁移 $t \in T$ 在状态 m 可执行且

$$\forall p \in P.(m'(p) = m(p) - \alpha_0(p, t) + \alpha_1(p, t)).$$

M 的迁移关系 $\to$ ，运行集合和可达状态集合的定义类似于普通 Petri 网模型。

Proposition 4.10 *每一个容量限制的 Petri 网模型都可以转换成一个等效的普通 Petri 网模型。*

§4.6 通信系统

通信系统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。

§4.6.1 通道

Definition 4.19 给定一个类型 $\langle \Lambda, n\rangle$ 其中 Λ 为字母表，n 为自然数，表示通道容量。一个类型为 $\langle \Lambda, n\rangle$ 的通道 m，记作 $m \in \langle \Lambda, n\rangle$，是一个值域为 $\bigcup_{i=0}^{n}\{\langle x_1, ..., x_i\rangle \mid x_i \in \Lambda\}$ 的变量。

通道状态记载通道的给定赋值，记为 σ。若 $m \in \langle \Lambda, n\rangle$，则 $\sigma(m) \in \bigcup_{i=0}^{n}\{\langle x_1, ..., x_i\rangle \mid x_i \in \Lambda\}$。通道状态空间为通道状态的集合，记为 Σ。

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

可执行事件: 对于 $x = \langle x_1, ..., x_n\rangle$，定义 $|x| = n$, $x \vdash s = \langle x_1, ..., x_n, s\rangle$, $HEAD(x) = x_1$, $TAIL(x) = \langle x_2, ..., x_n\rangle$。给定一个通道状态 σ 和一个通道 $m \in \langle \Lambda, n\rangle$。若以下一项成立则 $a \in \alpha(m) \cup \{\epsilon\}$ 可执行。

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

给定 $\sigma \in \Sigma$,$m \in \langle \Lambda, n\rangle$,$a \in \alpha(m)$。用 $\sigma \xrightarrow{a} \sigma'$ 表示 a 在 σ 可执行且以下成立。

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

§4.6.2 通信单元

Definition 4.20 一个通信单元是一个四元组 $\langle Q, C, \Delta, q_0\rangle$ 其中 Q 为状态的有穷集合，C 为通道的有穷集合，$\Delta \subseteq Q \times (\alpha(C) \cup \{\epsilon\}) \times Q$ 为标号的迁移关系，$q_0 \in Q$ 为初始状态。

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

迁移关系: 迁移 (q, a, q') 在 (s, σ) 可执行，当且仅当 $q = s$ 且 a 在 σ 可执行。系统状态的迁移关系 $\rightarrow$ 为 $(Q \times \Sigma)^2$ 的子集。$(q, \sigma) \rightarrow (q', \sigma')$ 当且仅当存在在状态 (q, σ) 的可执行迁移 (q, a, q') 使得 $\sigma \xrightarrow{a} \sigma'$ 或不存在可执行迁移且 $(q', \sigma') = (q, \sigma)$。

运行: 一个运行是 $Q \times \Sigma$ 上的一个无穷序列 $[(s_i, \sigma_i)]_{i \geq 0}$ 满足 $s_0 = q_0$, $\forall m \in C.(\sigma_0(m) = \langle\rangle)$ 且对任意 $i \geq 0$, $(s_i, \sigma_i) \rightarrow (s_{i+1}, \sigma_{i+1})$。通信单元 P 的运行集合记为 $[[P]]$。

§4.6.3 通信系统

为方便建摸，通常用一组通信单元构建通信模型。

Definition 4.21 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。

系统状态: 定义 $Q_1 \otimes \cdots \otimes Q_n = \{\{q_1, ..., q_n\} \mid q_1 \in Q_1, ..., q_n \in Q_n\}$ 。通信系统的系统状态的集合为 $(Q_1 \otimes \cdots \otimes Q_n) \times \Sigma$ 。

可执行迁移: 一个迁移 $(q, a, q') \in \Delta_1 \cup \cdots \cup \Delta_n$ 在状态 (Q, σ) 可执行当且仅当 $q \in Q$ 且 a 在 σ 可执行。

迁移关系: 给定 $a \in \alpha(C) \cup \{\epsilon\}$ 。 $(z, \sigma) \xrightarrow{a} (z', \sigma')$ 当且仅当存在在状态 (z, σ) 的可执行迁移 $(q, a, q') \in \Delta_1 \cup \cdots \cup \Delta_n$ 使得 $z' = (z \setminus \{q\}) \cup \{q'\}$ 且 $\sigma \xrightarrow{a} \sigma'$ 。 系统状态的迁移关系 $\rightarrow$ 为 $((Q_1 \otimes \cdots \otimes Q_n) \times \Sigma)^2$ 的子集。$(z, \sigma) \rightarrow (z', \sigma')$ 当且仅当存在在状态 (z, σ) 的可执行迁移 (q, a, q') 使得 $(z, \sigma) \xrightarrow{a} (z', \sigma')$ 或不存在可执行迁移且 $(z', \sigma') = (z, \sigma)$ 。

运行: 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 $[(z_i, \sigma_i)]_{i \geq 0}$ 满足 $z_0 = \{q_0, ..., q_n\}$, $\sigma_0(m_1) = \cdots = \sigma_0(m_k) = \langle\rangle$ 且对任意 $i \geq 0$, $(z_i, \sigma_i) \rightarrow (z_{i+1}, \sigma_{i+1})$ 。通信系统 M 的运行集合记为 $[[M]]$ 。

§4.6.4 通信系统与通信单元

由多个通信单元构建的通信系统并不比通信单元具有更强的表达能力。给定通信系统，我们可以构造行为等价的通信单元。

定义 $g(a)$ 如下。

若 a 不是一个集合，则 $g(a) = \{a\}$ 。若 $a = \{a_1, ..., a_j\}$ ，则 $g(a) = g(a_1) \cup \cdots \cup g(a_j)$ 。

定义 $g(a, b) = g(a) \cup g(b)$ 。

定义 $||a||$ 如下。

若 a 不是一个集合，则 $||a|| = 1$ 。若 $a = \{a_1, ..., a_j\}$ ，则 $||a|| = ||a_1|| + \cdots + ||a_j||$ 。

定义 $|x|$ 为集合 x 的元素个数。

给定一个集合 Q 。若 $\forall q \in Q.(|g(q)| = ||q||)$ ，则称 Q 为不可压缩集合。

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 和 Q_2 不相交且 $Q_1 \otimes Q_2$ 为不可压缩集合。

定义 $Q = \{g(q_1, q_2) \mid q_1 \in Q_1, q_2 \in Q_2\}$ 。

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

由 P_1 和 P_2 组合而成的通信单元用并发算子定义如下

$$P_1 || P_2 = \langle Q, C_1 \cup C_2, \Delta, g(q_{10}, q_{20})\rangle.$$

由以上定义容易看出并发算子满足结合率和交换率。

Proposition 4.11 *给定通信系统 $M = \{P_1, ..., P_n\}$ 且其中 $Q_1, ..., Q_n$ 为两两不相交且两两的 $\otimes$ 操作结果为不可压缩集合。定义通信单元 $P = P_1 || \cdots || P_n$ 。则 $[[P]] = [[M]]$ 。*

§4.6.5 通信单元与卫式迁移模型的等价

考虑一种关于通信单元运行中 Q 的序列的等价关系。

给定通信单元 $P = \langle Q, C, \Delta, q_0\rangle$ 。

给定 $B = (F, P)$ 和 V 且 $Q \subseteq F$ 和 $pc \in V$ 。

给定 (B, V) 上的卫式迁移模型 $M = (T, \Theta)$ 和 B 的解释 $I = (D, I_0)$ 。

给定 $\zeta : Q \rightarrow D$ 为单射函数。

Definition 4.22 P 与 M 是 ζ 计算等价的，当且仅当对任意 P 的计算 $[(s_i,\sigma_i)]_{i\geq 0}$ 存在 M 的计算 $[\sigma'_i]_{i\geq 0}$ 满足 $\forall i \geq 0.(\sigma'_i \models_I pc = s_i)$ 且对任意 M 的计算 $[\sigma'_i]_{i\geq 0}$ 存在 P 的计算 $[(\zeta^{-1}(\sigma'(pc)),\sigma_i)]_{i\geq 0}$ 。

通信单元的卫式迁移模型的构造: 给定通信单元 $P=\langle Q,C,\Delta,q_0\rangle$ 。设 $C=\{m_1,...,m_k\}$ 且 m_i 的类型为 $\langle \Lambda_i,n_i\rangle$ 。

定义 $B=(F,P)$ 其中 $F=\Lambda_1\cup\cdots\cup\Lambda_k\cup Q\cup\{0,1,+,-\}$ 且 $P=\{>,=,<\}$ 。

定义 $V=\{v_{i,j}\mid i\in\{1,...,k\},j\in\{0,...,n_i-1\}\}\cup\{pc\}$ 。

我们构造 (B,V) 上的卫式迁移模型 (T,Θ) 如下。

对每个 (q,a,q') 构造如下迁移并将其加入迁移集合 T 。

(1) 若 $a=\epsilon$ ，则构造迁移

$$pc=q \longrightarrow (pc):=(q').$$

(2) 若 $a=m_i!s$ ，则构造迁移

$$pc=q\wedge len_i<n_i \longrightarrow (v_{i,len_i},len_i,pc):=(s,len_i+1,q').$$

(3) 若 $a=m_i?s$ ，则构造迁移

$$pc=q\wedge len_i>0\wedge x_{i,0}=s \longrightarrow (v_{i,0},...,v_{i,len_i-1},len_i,pc):=(v_{i,1},...,v_{i,len_i},len_i-1,q').$$

定义 $\Theta=(pc=q_0\wedge\bigwedge_{i=1}^{k}(len_i=0\wedge\bigwedge_{j=1}^{n_i-1}v_{i,j}=0))$ 。

定义 $(P)^{gm}=(T,\Theta)$ 。则 $(P)^{gm}$ 是 (B,V) 上的卫式迁移模型。

给定 B 的解释 $I=(D,I_0)$ 其中 $D=N$ 为自然数集合， I_0 将 $\{0,1,+,-\}$ 解释为自然数上的常数及相应运算，将 $\Lambda_1\cup\cdots\cup\Lambda_k\cup Q$ 中的元素解释为自然数上的不同常数，将 $\{>,=,<\}$ 中的符号解释为自然数上的相应关系。

Proposition 4.12 设 $\zeta:Q\to D$ 的定义为 $\zeta(q)=I_0(q)$ 。则 $(P)^{gm}$ 与 P 是 ζ 计算等价的。

§4.7 练习

1. 定义泛 Büchi 自动机的交运算和并运算，即给定泛 Büchi 自动机 $\mathcal{B}_1=\langle\Sigma,S_1,\Delta_1,I_1,F_1\rangle$ 和 $\mathcal{B}_2=\langle\Sigma,S_2,\Delta_2,I_2,F_2\rangle$ ，定义 $B=B_1\cap B_2$ 使得 $\mathcal{L}(B)=\mathcal{L}(B_1)\cap\mathcal{L}(B_2)$ ；定义 $B=B_1\cup B_2$ 使得 $\mathcal{L}(B)=\mathcal{L}(B_1)\cup\mathcal{L}(B_2)$ 。

2. 用时间自动机描述交通灯的黄红绿变化。要求周期长度为 30 ，黄灯长度为 1 。 要求绿红两灯的长度差别最多为 10 且若两灯的长度差别不为零则长度差别逐渐减少。

3. 定义一种通信单元与标号 Kripke 结构的合理的等价关系和一种满足所定义等价关系的由通信单元构造标号 Kripke 结构的算法。

§5 线性时序逻辑

在命题逻辑和一阶逻辑的基础上增加模态算子用以描述时间先后次序有关的性质的一类逻辑称为时序逻辑。 本章介绍线性时序逻辑。

§5.1 命题线性时序逻辑 (PLTL)

考虑建立在命题逻辑上的线性时序逻辑，即命题线性时序逻辑，记作 PLTL 。 给定一个原子命题集合 AP 。用 p 表示 AP 中的任意命题。 PLTL 公式的集合由以下语法给出。

$$\phi ::= \quad p \mid \neg\phi \mid \phi \wedge \phi \mid \phi \vee \phi \mid \phi \to \phi \mid \phi \leftrightarrow \phi \mid O\,\phi \mid \Diamond\,\phi \mid \Box\,\phi \mid \phi\, U\, \phi \mid \phi\, R\, \phi$$

符号 $O, \Diamond, \Box, U, R$ 称为 PLTL 的时序算子。

PLTL 公式的语义： AP 上的 PLTL 公式在标号 Kripke 结构上解释。给定 $M = \langle S, R, I, L\rangle$ 。状态 $s \in S$ 满足 PLTL 公式 φ 记为 $M, s \models \varphi$ 当且仅当对所有 s 起点的无穷路径 π 有 $M, \pi \models \varphi$ 。用 π^k 来表示 π_k 开始的序列 $[\pi_i]_{i\geq k}$ 。 $M, \pi \models \varphi$ 定义如下。

$M, \pi \models p$	若 $p \in AP$ 且 $p \in L(\pi_0)$
$M, \pi \models \neg\varphi$	若 $M, \pi \not\models \varphi$
$M, \pi \models \varphi \vee \psi$	若 $M, \pi \models \varphi$ 或 $M, \pi \models \psi$
$M, \pi \models \varphi \wedge \psi$	若 $M, \pi \models \varphi$ 且 $M, \pi \models \psi$
$M, \pi \models \varphi \to \psi$	若 $M, \pi \models \varphi$ 则 $M, \pi \models \psi$
$M, \pi \models \varphi \leftrightarrow \psi$	若 $M, \pi \models \varphi \to \psi$ 且 $M, \pi \models \psi \to \varphi$
$M, \pi \models O\varphi$	若 $M, \pi^1 \models \varphi$
$M, \pi \models \Box\psi$	若 $\forall i \geq 0, M, \pi^i \models \psi$
$M, \pi \models \Diamond\varphi$	若 $\exists i \geq 0, M, \pi^i \models \varphi$
$M, \pi \models \varphi U \psi$	若 $\exists i \geq 0.(M, \pi^i \models \psi$ 且 $\forall 0 \leq j < i, M, \pi^j \models \varphi)$
$M, \pi \models \varphi R \psi$	若 $\forall i \geq 0.(M, \pi^i \models \psi$ 或 $\exists 0 \leq j < i, M, \pi^j \models \varphi)$

Definition 5.1 $M \models \varphi$ 当且仅当对所有 $s \in I$ 有 $M, s \models \varphi$ 。

可满足公式: 设 φ 为 PLTL 公式。 φ 是可满足的，当且仅当存在标号 Kripke 结构 M 使得 $M \models \varphi$ 。

重言式和等价: 设 φ 和 ψ 为 PLTL 公式。 φ 是重言式，记作 $\models \varphi$ ，当且仅当对任意标号 Kripke 结构 M 有 $M \models \varphi$ 。 φ 等价于 ψ ，记作 $\varphi \equiv \psi$ ，当且仅当 $\models \varphi \leftrightarrow \psi$ 。

等价公式: 为方便起见，定义 $\top = (p_0 \vee \neg p_0)$ 和 $\bot = (p_0 \wedge \neg p_0)$ 其中 $p_0 \in AP$ 为给定命题。设 φ 和 ψ 为 PLTL 公式。我们有以下等价的公式对。

1.	$O\varphi$	$\equiv$	$\neg(O\neg\varphi)$
2.	$\Box\varphi$	$\equiv$	$\neg(\Diamond\neg\varphi)$
3.	$\varphi R\psi$	$\equiv$	$\neg(\neg\varphi U\neg\psi)$
4.	$\Box\varphi$	$\equiv$	$(\varphi \wedge O(\Box\varphi))$
5.	$\Diamond\varphi$	$\equiv$	$(\varphi \vee O(\Diamond\varphi))$
6.	$\varphi R\psi$	$\equiv$	$(\psi \wedge (\varphi \vee O(\varphi R\psi)))$
7.	$\varphi U\psi$	$\equiv$	$(\psi \vee (\varphi \wedge O(\varphi U\psi)))$
8.	$\Diamond\varphi$	$\equiv$	$(\top U\varphi)$
9.	$\Box\varphi$	$\equiv$	$(\bot R\varphi)$
10.	$\varphi R\psi$	$\equiv$	$(\psi U(\varphi \wedge \psi) \vee G\psi)$

完全集: 一个逻辑联接符和时序算子的集合 Y 称为 PLTL 的完全集，当且仅当每个 PLTL 公式都等价于一个联接符和时序算子都在 Y 中的 PLTL 公式。 一个时序算子的集合 Y 称为 PLTL 的时序算子完全集，当且仅当每个 PLTL 公式都等价于一个时序算子在 Y 中的 PLTL 公式。

极小完全集: 一个逻辑联接符和时序算子的集合 Y 称为 PLTL 的极小完全集，当且仅当 Y 是 PLTL 的完全集且 Y 的真子集都不是 PLTL 的完全集。 一个时序算子的集合 Y 称为 PLTL 的时序算子极小完全集，当且仅当 Y 是 PLTL 的时序算子完全集且 Y 的真子集都不是 PLTL 的时序算子完全集。

Proposition 5.1 $\{O, U\}$ 是 *PLTL* 的时序算子极小完全集。

由以上等价的公式类型，我们知道 $\{O, U\}$ 是 PLTL 的时序算子完全集。设 $AP = \{p\}$ 和 $\varphi = (\neg pUp)$ 。我们可以构造两个集合 $A \subseteq B \subseteq (2^{AP})^\omega$ 使得 A 的所有元素都满足 φ 且 B 中有元素不满足 φ 但不存在任何只使用时序算子 O 的公式具有同样性质。类似地，我们可以证明 $\{U\}$ 不是 PLTL 的时序算子完全集。

Corollary 5.1 $\{O, U, \neg, \vee\}$ 是 *PLTL* 的极小完全集。

NNF 范式: 只使用逻辑连接符 $\wedge, \vee, \neg$ 且逻辑联接符 $\neg$ 只出现在命题前面的公式称为 NNF 范式。

Proposition 5.2 每个 *PLTL* 公式等价于一个 *PLTL* 的 *NNF* 范式。

对于每个 PLTL 公式，我们首先可以将其转换成等价的只使用逻辑联接符和时序算子在 $\{O, U, \neg, \vee\}$ 中的公式.然后使用以下等价关系将其转换成逻辑联接符和时序算子在 $\{O, U, R, \neg, \vee, \wedge\}$ 中的 NNF 公式。

$\neg\neg\varphi$	$\equiv$	φ
$\neg(\varphi \vee \psi)$	$\equiv$	$\neg\varphi \wedge \neg\psi$
$\neg(\varphi \wedge \psi)$	$\equiv$	$\neg\varphi \vee \neg\psi$
$\neg(O\varphi)$	$\equiv$	$O(\neg\varphi)$
$\neg(\varphi U\psi)$	$\equiv$	$\neg\varphi R\neg\psi$
$\neg(\varphi R\psi)$	$\equiv$	$\neg\varphi U\neg\psi$

可满足性判定问题: 判定 PLTL 公式是否可满足的问题称为 PLTL 可满足性问题。 PLTL 可满足性问题的复杂性为 PSPACE 完全。 只允许时序算子 $\Box$ 和 $\Diamond$ 的 PLTL 公式的子集记为 PLTL(F) 。 PLTL(F) 可满足性问题的复杂性为 NP 完全。

Proposition 5.3 *给定 PLTL 公式 φ 。若 φ 可满足，则存在 M 和 $\pi = \pi_0 \cdots \pi_{i-1}(\pi_i \cdots \pi_k)^\omega$ 使得 $M, \pi \models \varphi$ 且 $k \leq 2^{2|\varphi|}$ 。*

模型检测问题: 判定模型是否满足 PLTL 公式的问题称为 PLTL 模型检测问题。 PLTL 模型检测问题的复杂性为 PSPACE 完全。

§5.1.1 PLTL 公式的推理

以 $\{O, U, \Box, \Diamond\}$ 为 PLTL 公式的时序算子集。PLTL 的推理系统包含以下三部分：一部分为 PLTL 公式相关的时序逻辑公理；另一部分为命题逻辑推理系统；第三部分为时序推理规则。设 φ 和 ψ 为 PLTL 公式。

时序逻辑公理:

$\Diamond\neg\neg\varphi \leftrightarrow \Diamond\varphi$
$\Box(\varphi \to \psi) \to (\Box\varphi \to \Box\psi)$
$\Box\varphi \to \varphi \wedge O\varphi \wedge O\Box\varphi$
$\Box(\varphi \to O\varphi) \to (\varphi \to \Box\varphi)$
$O\neg\varphi \leftrightarrow \neg O\varphi$
$O(\varphi \to \psi) \to (O\varphi \to O\psi)$
$(\varphi U \psi) \leftrightarrow (\psi \vee (\varphi \wedge O(\varphi U \psi))$
$(\varphi U \psi) \to \Diamond\psi$

命题逻辑推理系统:

如果 φ 是命题逻辑的重言式的实例，则 φ 是公理。
如果 $\vdash \varphi \to \psi$ 且 $\vdash \varphi$ ，则 $\vdash \psi$ 。

时序推理规则:

如果 $\vdash \varphi$ ，则 $\vdash \Box\varphi$ 。

Proposition 5.4 *PLTL 推理系统是可靠且完备的。*

§5.1.2 PLTL 限界语义

给定有穷状态标号 Kripke 结构 $M = \langle S, R, I, L\rangle$ 。称长度 $k+1$ 的路径为 k 路径。 k 路径 $[\pi_i]_{i=0}^k$ 称为 (k, ℓ) 环当且仅当 $R(\pi_k, \pi_\ell)$ 。

Definition 5.2 *设 φ 为 PLTL 公式。 $M, s \models^E \varphi$ 当且仅当存在 s 起点的无穷路径 π 使得 $M, \pi \models \varphi$ 。*

Proposition 5.5 *设 φ 为 PLTL 公式。 $M, s \models^E \varphi$ 当且仅当存在一个 s 起点的 (k, ℓ) 环使得 $M, \pi_0 \cdots \pi_{\ell-1}(\pi_\ell \cdots \pi_k)^\omega \models \varphi$ 且 $k \leq |M| \cdot 2^{2|\varphi|}$ 。*

以下考虑 NNF 范式的 PLTL 公式。设 π 为 k 路径。

非环的语义：定义 $M,\pi \models_{a,k} \varphi$ 为 $M,\pi \models^0_{a,k} \varphi$ 且后者的定义如下。

$M,\pi \models^i_{a,k} p$	若 $p \in AP$ 且 $p \in L(\pi_i)$
$M,\pi \models^i_{a,k} \neg p$	若 $M,\pi \not\models^i_{a,k} p$
$M,\pi \models^i_{a,k} \varphi \wedge \psi$	若 $M,\pi \models^i_{a,k} \varphi$ 且 $M,\pi \models^i_{a,k} \psi$
$M,\pi \models^i_{a,k} \varphi \vee \psi$	若 $M,\pi \models^i_{a,k} \varphi$ 或 $M,\pi \models^i_{a,k} \psi$
$M,\pi \models^i_{a,k} O\varphi$	若 $i < k$ 且 $M,\pi \models^{i+1}_{a,k} \varphi$
$M,\pi \models^i_{a,k} \Box\varphi$	若 $false$
$M,\pi \models^i_{a,k} \varphi U \psi$	若 $\bigvee_{j=i}^{k}((M,\pi \models^j_{a,k} \varphi) \wedge \bigwedge_{m=i}^{j-1}(M,\pi \models^m_{a,k} \varphi))$

Proposition 5.6 $M,\pi \models_{a,k} \varphi$ 当且仅当对任意 $\pi' \in \pi \cdot (S)^\omega$，$M,\pi' \models \varphi$。

环的语义：设 $0 \le \ell \le k$。定义 $M,\pi \models_{\ell,k} \varphi$ 为 $M,\pi \models^0_{\ell,k} \varphi$ 且后者的定义如下。

$M,\pi \models^i_{\ell,k} p$	若 $p \in AP$ 且 $p \in L(\pi_i)$
$M,\pi \models^i_{\ell,k} \neg p$	若 $M,\pi \not\models^i_{\ell,k} p$
$M,\pi \models^i_{\ell,k} \varphi \wedge \psi$	若 $M,\pi \models^i_{\ell,k} \varphi$ 且 $M,\pi \models^i_{\ell,k} \psi$
$M,\pi \models^i_{\ell,k} \varphi \vee \psi$	若 $M,\pi \models^i_{\ell,k} \varphi$ 或 $M,\pi \models^i_{\ell,k} \psi$
$M,\pi \models^i_{\ell,k} O\varphi$	若 $i < k$ 且 $M,\pi \models^{i+1}_{\ell,k} \varphi$ 或 $i = k$ 且 $M,\pi \models^{\ell}_{\ell,k} \varphi$
$M,\pi \models^i_{\ell,k} \Box\varphi$	若 $\bigwedge_{j=\min(i,\ell)}^{k}(M,\pi \models^j_{\ell,k} \varphi)$
$M,\pi \models^i_{\ell,k} \varphi U \psi$	若 $\bigvee_{j=i}^{k}((M,\pi \models^j_{\ell,k} \psi) \wedge \bigwedge_{m=i}^{j-1}(M,\pi \models^m_{\ell,k} \varphi)) \vee$ $\bigwedge_{j=i}^{k}(M,\pi \models^j_{\ell,k} \varphi) \wedge \bigvee_{j=\ell}^{i-1}((M,\pi \models^j_{\ell,k} \psi) \wedge \bigwedge_{m=i}^{j-1}(M,\pi \models^m_{\ell,k} \varphi))$

Proposition 5.7 $M,\pi \models_{\ell,k} \varphi$ 其中 $0 \le \ell \le k$，当且仅当 $M,\pi_0 \cdots \pi_{\ell-1}(\pi_\ell \cdots \pi_k)^\omega \models \varphi$。

限界语义：

Definition 5.3 $M,\pi \models_k \varphi$ 当且仅当 $M,\pi \models_{a,k} \varphi$ 或 $\bigvee_{i=0}^{k}(T(\pi_k,\pi_\ell) \wedge (M,\pi \models_{\ell,k} \varphi))$。

Definition 5.4 $M,s \models^E_k \varphi$，当且仅当存在 s 起点的 k 路径 π 使得 $M,\pi \models_k \varphi$。

正确性与完备性：

Proposition 5.8 若存在 $k \ge 0$ 使得 $M,s \models^E_k \varphi$，则 $M,s \models^E \varphi$。

Corollary 5.2 若存在 $k \ge 0$ 和 I 起点的 k 路径 π 使得 $M,\pi \models_k \neg\varphi$，则 $M \not\models \varphi$。

Proposition 5.9 若 $M,s \models^E \varphi$，则存在 $k \ge 0$ 使得 $M,s \models^E_k \varphi$。

Corollary 5.3 若对所有 $k \ge 0$ 和 I 起点的 k 路径 π 有 $M,\pi \not\models_k \neg\varphi$，则 $M \models \varphi$。

根据限界语义的定义，若存在 I 起点的 k 路径 π 使得 $M,\pi \models_k \varphi$，则对所有 $i \ge 0$, 存在 I 起点的 k 路径 π 使得 $M,\pi \models_{k+i} \varphi$。

定义 (M,φ) 的完备阈值 $ct(M,\varphi)$ 为满足以下条件最小的 k。

$$\forall s \in I.(M,s \not\models^E_k \neg\varphi)$$，则对所有 $i \ge 0, \forall s \in I.(M,s \not\models^E_{k+i} \neg\varphi)$。

对任何 M 和 φ，完备阈值 $ct(M,\varphi)$ 存在且 $ct(M,\varphi) \le |M| \cdot 2^{2|\varphi|}$。进而我们有以下结论。

Corollary 5.4 若 $k \ge ct(M,\varphi)$ 且对所有 I 起点的 k 路径 π 有 $M,\pi \not\models_k \neg\varphi$，则 $M \models \varphi$。

§5.2 PLTL 公式的不动点表示与线性 μ 演算 (νTL)

线性结构: 给定原子命题集合 AP 。一个线性结构为三元组 $\langle S, \zeta, L\rangle$ ，其中 S 为状态集合，$\zeta = [\zeta_i]_{i\geq 0} \in S^\omega$ 为 S 的无穷序列， $L : S \to 2^{AP}$ 为标号函数。

Definition 5.5 *给定线性结构 $\langle S, \zeta, L\rangle$ 和 PLTL 公式 φ 。构造标号 Kripke 模型 $M = \langle S, S \times S, S, L\rangle$ 。 $\langle S, \zeta, L\rangle \models \varphi$ 当且仅当 $M, \zeta \models \varphi$ 。*

将 $\mathbf{N}$ 的子集 $\{i \mid \langle S, \zeta^i, L\rangle \models \varphi\}$ 记为 $[[\varphi]]$ 。定义函数 $o : 2^{\mathbf{N}} \to 2^{\mathbf{N}}$ 如下:

$$o(A) = \{i \in \mathbf{N} \mid i + 1 \in A\}$$

我们有以下等式:

$[[p]]$	$=$	$\{i \mid p \in L(\zeta_i)\}$
$[[\neg\varphi]]$	$=$	$\mathbf{N} \setminus [[\varphi]]$
$[[\varphi \vee \psi]]$	$=$	$[[\varphi]] \cup [[\psi]]$
$[[O\varphi]]$	$=$	$o([[\varphi]])$
$[[\varphi U \psi]]$	$=$	$[[\psi]] \cup ([[\varphi]] \cap [[O(\varphi U \psi)]]$

由以上等式知 $[[\varphi U\psi]]$ 是 $\tau(Z) = [[\psi]] \cup ([[\varphi]] \cap o(Z))$ 的不动点。 根据 τ 的定义知 τ 有最小和最大不动点。 根据 PLTL 语义知 $[[\varphi U\psi]]$ 是 τ 的最小不动点，即

$$[[\varphi U\psi]] \quad = \quad \mu Z.([[\psi]] \cup ([[\varphi]] \cap o(Z))$$

类似地，其它时序算子也可以用不动点刻画。为方便书写，我们直接将 φ, ψ 看成 $\mathbf{N}$ 的子集，时序算子和逻辑联接符看成是 $2^{\mathbf{N}}$ 上的函数。

$\Diamond\varphi$	$=$	$\mu Z.(\varphi \vee OZ)$
$\Box\varphi$	$=$	$\nu Z.(\varphi \wedge OZ)$
$\varphi U \psi$	$=$	$\mu Z.(\psi \vee (\varphi \wedge OZ))$
$\varphi R \psi$	$=$	$\nu Z.(\psi \wedge (\varphi \vee OZ))$

Proposition 5.10 *给定线性结构 $\langle S, \zeta, L\rangle$ 和 PLTL 公式 φ 。$\langle S, \zeta, L\rangle \models \varphi$ 当且仅当 $0 \in [[\varphi]]$ 。*

线性 μ 演算: 由关于 PLTL 的讨论我们知道 PLTL 公式可以用时序算子 O 和不动点表示。在这种表示中，命题变量只出现在 O, μ, ν 之后。将这个约束去掉，我们可以得到一个表达能力更强的逻辑。称为 νTL 。 给定一个原子命题集合 AP ，一个变量集合 V 。用 p 表示 AP 中任意命题， X 表示 V 中任意变量。 νTL 公式的集合由以下语法给出。

$$\phi ::= p \mid X \mid \neg\phi \mid \phi \wedge \phi \mid \phi \vee \phi \mid O\phi \mid \mu X.\phi \mid \nu X.\phi$$

其中 $\mu X.\phi$ 和 $\nu X.\phi$ 的 ϕ 中不受囿的 X 必须在偶数个 $\neg$ 符号的作用范围之下。

νTL 公式在线性结构上的语义解释: 给定线性结构 $\langle S, \zeta, L\rangle$ 和 νTL 公式 φ 。 设 $e : V \to 2^N$ 为

变量到 N 子集的赋值。 νTL 公式的语义如下：

$$
\begin{array}{lcl}
[[p]]e & = & \{i \in N \mid p \in L(\zeta_i)\} \\
[[X]]e & = & e(X) \\
[[\neg\phi]]e & = & N \setminus [[\phi]]e \\
[[\phi_1 \wedge \phi_2]]e & = & [[\phi_1]]e \cap [[\phi_2]]e \\
[[\phi_1 \vee \phi_2]]e & = & [[\phi_1]]e \cup [[\phi_2]]e \\
[[O\phi]]e & = & \{i \in N \mid i+1 \in [[\phi]]e\} \\
[[\mu X.\phi]]e & = & \cap\{U \subseteq N \mid [[\phi]]e[X/U] \subseteq U\} \\
[[\nu X.\phi]]e & = & \cup\{U \subseteq N \mid U \subseteq [[\phi]]e[X/U]\}
\end{array}
$$

Definition 5.6 *给定线性结构 $\langle S, \zeta, L\rangle$ 和 νTL 公式 φ 。 $\langle S, \zeta, L\rangle \models \varphi$ 当且仅当 $0 \in [[\varphi]]$ 。*

可满足公式: 设 φ 为 νTL 公式。 φ 是可满足的，当且仅当存在 $\langle S, \zeta, L\rangle$ 使得 $\langle S, \zeta, L\rangle \models \varphi$ 。

重言式和等价: 设 φ 为 νTL 公式。 φ 是重言式，记作 $\models \varphi$ ，当且仅当对任意 $\langle S, \zeta, L\rangle$ 有 $\langle S, \zeta, L\rangle \models \varphi$ 。 φ 等价于 ψ ，记作 $\varphi \equiv \psi$ ，当且仅当 $\models (\varphi \wedge \psi) \vee (\neg\varphi \wedge \neg\psi)$ 。

模态算子的对偶关系与 NNF 范式: 只使用逻辑连接符 $\wedge, \vee, \neg$ 且逻辑连接符 $\neg$ 只出现在命题前面的公式称为 NNF 范式。每个 νTL 公式等价于一个 νTL 的 NNF 范式。 一个 νTL 的 NNF 范式可应用以下对偶关系构造。

$$
\begin{array}{lcl}
\neg(O p) & \equiv & O\neg p \\
\neg\mu X.\phi & \equiv & \nu X.\neg\phi(\neg X/X)
\end{array}
$$

PLTL 公式到 νTL 公式的转换: PLTL 公式可以看成是 νTL 的一个子类，可以用以下规则将 PLTL 公式转换到 νTL 公式。

$$
\begin{array}{lcl}
T(p) & = & p \\
T(\neg\varphi) & = & \neg T(\varphi) \\
T(\varphi \vee \psi) & = & T(\varphi) \vee T(\psi) \\
T(O\varphi) & = & OT(\varphi) \\
T(\varphi U \psi) & = & \mu Y.(T(\psi) \vee (T(\varphi) \wedge OY))
\end{array}
$$

Proposition 5.11 *给定线性结构 $\langle S, \zeta, L\rangle$ 和 $PLTL$ 公式 φ 。 $\langle S, \zeta, L\rangle \models \varphi$ 当且仅当 $\langle S, \zeta, L\rangle \models T(\varphi)$ 。*

Proposition 5.12 *νTL 的表达能力强于 $PLTL$ 。*

可满足性判定问题: 判定 νTL 公式是否可满足的问题复杂性为 EXPTIME 完全。

模型检测问题: 判定模型是否满足 νTL 公式的问题复杂性为 PSPACE 完全。

§5.3 一阶线性时序逻辑

一阶线性时序逻辑用一阶逻辑来描述状态性质，具有表达能力强的优点。 给定 $B = (F, P)$ 上的一阶逻辑。用 p 表示 WFF_B 中的任意公式。 B 上的一阶线性时序逻辑公式的集合 LTL 由以下语法给出。

$$\phi ::= \quad p \mid \neg\phi \mid \phi \wedge \phi \mid \phi \vee \phi \mid \phi \rightarrow \phi \mid \phi \leftrightarrow \phi \mid O\,\phi \mid \Diamond\,\phi \mid \Box\,\phi \mid \phi\,U\,\phi \mid \phi\,R\,\phi$$

给定 B 的解释 I 。设变量赋值的集合为 Σ 。

模型: LTL 公式的模型 M 为满足以下条件的 Σ^ω 的非空子集。

若 $\zeta, \zeta' \in M$ 且 $\zeta_i = \zeta'_j$,	则 $\zeta_0 \cdots \zeta_i \zeta'_{j+1} \zeta'_{j+2} \cdots \in M$
若 $\zeta \in M$ 且 $\zeta_i = \zeta_j$ 且 $i < j$,	则 $\zeta_0 \cdots \zeta_{i-1} (\zeta_i \cdots \zeta_{j-1})^\omega \in M$

语义: 设 $\zeta = \zeta_0 \zeta_1 \zeta_2 \cdots \in \Sigma^\omega$ 。 ζ 满足 φ ，记作 $\zeta \models \varphi$ ，定义如下。

$\zeta \models p$	若 $p \in WFF_B$ 且 $\zeta_0 \models_I p$
$\zeta \models \neg\varphi$	若 $\zeta \not\models \varphi$
$\zeta \models \varphi \vee \psi$	若 $\zeta \models \varphi$ 或 $\zeta \models \psi$
$\zeta \models \varphi \wedge \psi$	若 $\zeta \models \varphi$ 且 $\zeta \models \psi$
$\zeta \models \varphi \to \psi$	若 $\zeta \models \varphi$ 则 $\zeta \models \psi$
$\zeta \models \varphi \leftrightarrow \psi$	若 $\zeta \models \varphi \to \psi$ 且 $\zeta \models \psi \to \varphi$
$\zeta \models O\varphi$	若 $\zeta^1 \models \varphi$
$\zeta \models \Box\psi$	若 $\forall i \geq 0, \zeta^i \models \psi$
$\zeta \models \Diamond\varphi$	若 $\exists i \geq 0, \zeta^i \models \varphi$
$\zeta \models \varphi U \psi$	若 $\exists i \geq 0.(\zeta^i \models \psi$ 且 $\forall 0 \leq j < i, \zeta^j \models \varphi)$
$\zeta \models \varphi R \psi$	若 $\forall i \geq 0.(\zeta^i \models \psi$ 或 $\exists 0 \leq j < i, \zeta^j \models \varphi)$

Definition 5.7 $M \models \varphi$ ，当且仅当对所有 $\zeta \in M$ 都有 $\zeta \models \varphi$ 。

可满足公式: 设 φ 为 LTL 公式。 φ 是可满足的，当且仅当存在 M 使得 $M \models \varphi$ 。

重言式和等价: 设 φ 和 ψ 为 LTL 公式。φ 是重言式, 记作 $\models \varphi$, 当且仅当对任意 M 有 $M \models \varphi$ 。φ 等价于 ψ ，记作 $\varphi \equiv \psi$ ，当且仅当 $\models \varphi \leftrightarrow \psi$ 。

完全集: 一阶线性时序逻辑中时序算子相互之间的可表达性和命题线性时序逻辑类似。我们有以下结论。

Proposition 5.13 $\{O, U\}$ 是 *LTL* 的时序算子极小完全集。

推理规则: 用 $\varphi \Rightarrow \psi$ 表示 $M \models \Box(\varphi \to \psi)$ 。 设 $e \in T_B$ 为项， $\sqsubseteq$ 为 P 中二元谓词符号, $w \in QFF_B$ 为一元谓词公式 (记其变量为 x) 且 $W = (\{\sigma(x) \mid I(w)(\sigma) = true\}, I_0(\sqsubseteq))$ 为良基集合， v 为没有在公式中出现过的变量。 根据 LTL 公式的语义， 我们有以下推理规则。

$$\frac{\varphi \Rightarrow \psi \qquad \psi \wedge \neg\varphi_0 \Rightarrow O\psi}{\varphi \Rightarrow \varphi_0 R \psi} \qquad \frac{\varphi \Rightarrow \psi \qquad \psi \Rightarrow O\psi}{\varphi \Rightarrow \Box\psi}$$

$$\frac{\varphi \Rightarrow \psi}{\varphi \Rightarrow \varphi U \psi} \qquad \frac{\varphi \Rightarrow \psi}{\varphi \Rightarrow \Diamond\psi}$$

$$\frac{\varphi \Rightarrow O\psi}{\varphi \Rightarrow \varphi U \psi} \qquad \frac{\varphi \Rightarrow O\psi}{\varphi \Rightarrow \Diamond\psi}$$

$$\frac{\varphi \Rightarrow (\psi \vee \phi) \qquad \phi \Rightarrow w_x^e \qquad \phi \wedge e = v \Rightarrow \varphi_0 U(\psi \vee (\phi \wedge e \sqsubset v))}{\varphi \Rightarrow \varphi_0 U \psi} \qquad \frac{\varphi \Rightarrow (\psi \vee \phi) \qquad \phi \Rightarrow w_x^e \qquad \phi \wedge e = v \Rightarrow \Diamond(\psi \vee (\phi \wedge e \sqsubset v))}{\varphi \Rightarrow \Diamond\psi}$$

§5.4 练习

1. 用 PLTL 写信号灯变化的规范：信号灯依次序黄红绿变化，每个状态有且只有一个信号，初始信号为黄色，黄色只停留一个状态，红绿色可以连续在多个状态上成立。

2. 设一杯茶售价 3 元。设计一个可用 1 元和 5 元硬币的自动售茶机的双标号迁移模型 M 。要求状态上的标号包括 q_0, q_1 ，其中 q_0 代表系统处在该状态时可投币， q_1 代表在该状态可取茶。若投入钱数超过 3 元，则需在退还多余的钱之后的状态取茶。由双标号迁移模型构造标号 Kripke 模型 M' 。用限界语义验证 M' 不满足 $A(q_0 U q_1)$ 并给出最小的可以确定以上公式不满足的界。

§6 分枝时序逻辑

本章介绍可用以描述程序模型性质的分枝时序逻辑。 抽象模型中的一个状态通常有多个可能的后继状态和可能从该状态出发的路径。线性时序逻辑仅表达线性运行过程的性质。表达这类分枝路径的性质则需要在逻辑中添加路径量词。

§6.1 计算树逻辑 (CTL)

考虑建立在命题逻辑上的计算树逻辑，记作 CTL 。 给定一个原子命题集合 AP 。用 p 表示 AP 中的任意命题。 PLTL 公式的集合由以下语法给出。

$$\begin{aligned}\phi ::= \quad & p \mid \neg\phi \mid \phi \wedge \phi \mid \phi \vee \phi \mid \phi \to \phi \mid \phi \leftrightarrow \phi \mid \\ & EX\,\phi \mid EF\,\phi \mid EG\,\phi \mid E(\phi\, U\, \phi) \mid E(\phi\, R\, \phi) \mid \\ & AX\,\phi \mid AF\,\phi \mid AG\,\phi \mid A(\phi\, U\, \phi) \mid A(\phi\, R\, \phi)\end{aligned}$$

符号 $EX, EF, EG, EU, ER, AX, AF, AG, AU, AR$ 称为 CTL 的时序算子。

CTL 公式的语义： AP 上的 CTL 公式在标号 Kripke 结构上解释。给定 $M = \langle S, R, I, L\rangle$ 。状态 $s \in S$ 满足 CTL 公式 φ ，记为 $M, s \models \varphi$ ，在 M 已确定的情况亦写做 $s \models \varphi$ ，定义如下。

$s \models p$	若 $p \in AP$ 且 $p \in L(s)$
$s \models \neg\varphi$	若 $s \not\models \varphi$
$s \models \varphi \vee \psi$	若 $s \models \varphi$ 或 $s \models \psi$
$s \models \varphi \wedge \psi$	若 $s \models \varphi$ 且 $s \models \psi$
$s \models \varphi \to \psi$	若 $s \models \varphi$ 则 $s \models \psi$
$s \models \varphi \leftrightarrow \psi$	若 $s \models \varphi \to \psi$ 且 $s \models \psi \to \varphi$
$s \models EX\varphi$	若存在 s 的后继 v 使得 $v \models \varphi$
$s \models EG\psi$	若存在 s 为起点的无穷路径 ζ 使得 $\forall i \geq 0, \zeta_i \models \psi$
$s \models EF\psi$	若存在 s 为起点的无穷路径 ζ 使得 $\exists i \geq 0, \zeta_i \models \psi$
$s \models E(\varphi U \psi)$	若存在 s 为起点的无穷路径 ζ 使得 $\exists i \geq 0.(\zeta_i \models \psi$ 且 $\forall 0 \leq j < i, \zeta_j \models \varphi)$
$s \models E(\varphi R \psi)$	若存在 s 为起点的无穷路径 ζ 使得 $\forall i \geq 0.(\zeta_i \models \psi$ 或 $\exists 0 \leq j < i, \zeta_j \models \varphi)$
$s \models AX\varphi$	若对所有 s 的后继 v 有 $v \models \varphi$
$s \models AG\psi$	若对所有 s 为起点的无穷路径 ζ 有 $\forall i \geq 0, \zeta_i \models \psi$
$s \models AF\psi$	若对所有 s 为起点的无穷路径 ζ 有 $\exists i \geq 0, \zeta_i \models \psi$
$s \models A(\varphi U \psi)$	若对所有 s 为起点的无穷路径 ζ 有 $\exists i \geq 0.(\zeta_i \models \psi$ 且 $\forall 0 \leq j < i, \zeta_j \models \varphi)$
$s \models A(\varphi R \psi)$	若对所有 s 为起点的无穷路径 ζ 有 $\forall i \geq 0.(\zeta_i \models \psi$ 或 $\exists 0 \leq j < i, \zeta_j \models \varphi)$

Definition 6.1 $M \models \varphi$ 当且仅当对所有 $s \in I$ 有 $M, s \models \varphi$ 。

可满足公式: 设 φ 为 CTL 公式。φ 是可满足的, 当且仅当存在标号 Kripke 结构 M 使得 $M \models \varphi$ 。

重言式和等价: 设 φ 和 ψ 为 CTL 公式。 φ 是重言式，记作 $\models \varphi$ ，当且仅当对标号 Kripke 结构 M 有 $M \models \varphi$ 。 φ 等价于 ψ ，记作 $\varphi \equiv \psi$ ，当且仅当 $\models \varphi \leftrightarrow \psi$ 。

等价公式: 设 φ 和 ψ 为 CTL 公式。我们有以下等价的公式对。

$AX\varphi$	$\equiv$	$\neg EX\neg\varphi$
$AG\varphi$	$\equiv$	$\neg EF\neg\varphi$
$AF\varphi$	$\equiv$	$\neg EG\neg\varphi$
$A(\varphi R\psi)$	$\equiv$	$\neg E(\neg\varphi U\neg\psi)$
$A(\varphi U\psi)$	$\equiv$	$\neg E(\neg\varphi R\neg\psi)$
$E(\varphi U\psi)$	$\equiv$	$\psi\vee(\varphi\wedge EXE(\psi U(\varphi\wedge\psi)))$
$E(\varphi R\psi)$	$\equiv$	$\psi\wedge(\varphi\vee EXE(\psi U(\varphi\wedge\psi)))$
$EF\varphi$	$\equiv$	$E(\top U\varphi)$
$EG\varphi$	$\equiv$	$E(\neg\top R\varphi)$
$E(\varphi R\psi)$	$\equiv$	$E(\psi U(\varphi\wedge\psi))\vee EG\psi$

Proposition 6.1 $\{EX, EG, EU\}$ 是 *CTL* 的时序算子极小完全集。

Corollary 6.1 $\{EX, EG, EU, \neg, \vee\}$ 是 *CTL* 的极小完全集。

NNF 范式: 只使用逻辑连接符 $\wedge, \vee, \neg$ 且逻辑联接符 $\neg$ 只出现在命题前面的公式称为 NNF 范式。

Proposition 6.2 每个 *CTL* 公式等价于一个 *CTL* 的 *NNF* 范式。

对于每个 CTL 公式，我们首先可以将其转换成等价的只使用逻辑联接符和时序算子在 $\{EX, EG, EU, \neg, \vee\}$ 中的公式，然后再继续使用以上提到的等价关系和以下与命题逻辑联接符相关的等价关系将其转换成逻辑联接符和时序算子在 $\{EX, ER, EU, AX, AR, AU, \neg, \vee, \wedge\}$ 中的 NNF 公式。

$\neg\neg\varphi$	$\equiv$	φ
$\neg(\varphi\vee\psi)$	$\equiv$	$\neg\varphi\wedge\neg\psi$
$\neg(\varphi\wedge\psi)$	$\equiv$	$\neg\varphi\vee\neg\psi$

可满足性判定问题: 判定 CTL 公式是否可满足的问题称为 CTL 可满足性问题。 CTL 可满足性问题的复杂性为 EXPTIME 完全。

模型检测问题: 判定模型是否满足 CTL 公式的问题称为 CTL 模型检测问题。 CTL 模型检测问题的复杂性为 P 完全。

§6.1.1 CTL 公式的推理

以 $\{EX, EU, EG, EF, AX, AU, AG, AF, \}$ 为 CTL 公式的时序算子集。 CTL 的推理系统包含以下三部分：一部分为 CTL 式相关的时序逻辑公理；另一部分为命题逻辑的推理系统；第三部分为时序推理规则。设 p, q, r 为 CTL 公式。

时序逻辑公理:

$EFp \leftrightarrow E(\top Up)$
$AGp \leftrightarrow \neg EF\neg p$
$AFp \leftrightarrow A(\top Up)$
$EGp \leftrightarrow \neg AF\neg p$
$EX(p \vee q) \leftrightarrow EXp \vee EXq$
$AXp \leftrightarrow \neg EX\neg p$
$E(pUq) \leftrightarrow (q \vee (p \wedge EXE(pUq))$
$A(pUq) \leftrightarrow (q \vee (p \wedge AXA(pUq))$
$EX\top \wedge AX\top$
$AG(r \rightarrow (\neg q \wedge EXr)) \rightarrow (r \rightarrow \neg A(pUq))$
$AG(r \rightarrow (\neg q \wedge EXr)) \rightarrow (r \rightarrow \neg AFq)$
$AG(r \rightarrow (\neg q \wedge (p \rightarrow AXr))) \rightarrow (r \rightarrow \neg E(pUq))$
$AG(r \rightarrow (\neg q \wedge AXr)) \rightarrow (r \rightarrow \neg EFq)$
$AG(p \rightarrow q) \rightarrow (EXp \rightarrow EXq)$

命题逻辑推理系统:

如果 p 是命题逻辑的重言式的实例，则 p 是公理。

如果 $\vdash p \rightarrow q$ 且 $\vdash p$ ，则 $\vdash q$ 。

时序推理规则:

如果 $\vdash p$ ，则 $\vdash AGp$ 。

Proposition 6.3 *CTL 推理系统是可靠且完备的。*

§6.1.2 CTL 限界语义

给定有穷状态标号 Kripke 结构 $M = \langle S, R, I, L\rangle$ 。称长度 $k+1$ 的路径为 k 路径。记 k 路径的集合为 P_k 。设 $\pi \in P_k$ 。定义

$$\gamma(\pi) = \bigvee_{i=0}^{|\pi|-2} \bigvee_{j=i+1}^{|\pi|-1} \pi_i = \pi_j$$

用 $\pi(s) \in P_k$ 表示 π 是 s 为起点的 k 路径。NNF 范式的 CTL 公式 φ 的限界语义 $M, s \models_k \varphi$ 定义如下:

$M, s \models_k p$	若 $p \in AP$ 且 $p \in L(s)$
$M, s \models_k \neg p$	若 $M, s \not\models_k p$
$M, s \models_k \varphi \vee \psi$	若 $M, s \models_k \varphi$ 或 $M, s \models_k \psi$
$M, s \models_k \varphi \wedge \psi$	若 $M, s \models_k \varphi$ 且 $M, s \models_k \psi$
$M, s \models_k AX\varphi$	若 $\forall \pi(s) \in P_k$ ： $M, \pi \models_k \varphi$
$M, s \models_k A(\varphi U \psi)$	若 $\forall \pi(s) \in P_k$ ： $\exists i \le k, M, \pi_i \models_k \psi$ 且 $\forall j < i, M, \pi_j \models_k \varphi$
$M, s \models_k A(\varphi R \psi)$	若 $\forall \pi(s) \in P_k$ ： $\forall i \le k$, 若 $\forall j < i, M, \pi_j \not\models_k \varphi$ 则 $M, \pi_i \models_k \psi$ ，且若 $\forall i \le k, M, \pi_i \not\models_k \varphi$ 则 $\gamma(\pi)$
$M, s \models_k EX\varphi$	若 $\exists \pi(s) \in P_k$ ： $M, \pi \models_k \varphi$
$M, s \models_k E(\varphi U \psi)$	若 $\exists \pi(s) \in P_k$ ： $\exists i \le k, M, \pi_i \models_k \psi$ 且 $\forall j < i, M, \pi_j \models_k \varphi$
$M, s \models_k E(\varphi R \psi)$	若 $\exists \pi(s) \in P_k$ ： $\forall i \le k$, 若 $\forall j < i, M, \pi_j \not\models_k \varphi$ 则 $M, \pi_i \models_k \psi$ ，且若 $\forall i \le k, M, \pi_i \not\models_k \varphi$ 则 $\gamma(\pi)$

Definition 6.2 $M \models_k \varphi$，当且仅当对所有 $s \in I$， $M, s \models_k \varphi$。

正确性与完备性：

Proposition 6.4 给定 M 和 φ。若存在 $k \geq 0$ 使得 $M \models_k \varphi$，则 $M \models \varphi$。

Proposition 6.5 给定 M 和 φ。若 $M \models \varphi$，则存在 $k \geq 0$ 使得 $M \models_k \varphi$。

§6.2 CTL 公式的不动点表示与模态 μ 演算

给定一个 AP 上的 Kripke 结构 $M = \langle S, R, I, L\rangle$。

给定的一个 CTL 公式 p。S 的子集 $\{s \in S \mid M, s \models p\}$ 记为 $[[p]]$。定义函数 $ex : 2^S \to 2^S$ 如下：

$$ex(A) = \{s \in S \mid \exists s' \in S.((s, s') \in R \wedge s' \in A)\}$$

我们有以下等式：

$[[p]]$	$=$	$\{s \mid p \in L(s)\}$
$[[\neg p]]$	$=$	$S \setminus [[p]]$
$[[p \vee q]]$	$=$	$[[p]] \cup [[q]]$
$[[EXp]]$	$=$	$ex([[p]])$
$[[EGp]]$	$=$	$[[p]] \cap ex([[EGp]])$
$[[E(pUq)]]$	$=$	$[[q]] \cup ([[p]] \cap ex([[E(pUq)]])$

由以上等式知 $[[EGp]]$ 和 $[[E(pUq)]]$ 分别是 $\tau_1(Z) = [[p]] \cap ex(Z)$ 和 $\tau_2(Z) = [[q]] \cup ([[p]] \cap ex(Z))$ 的不动点。根据 τ_1 和 τ_2 的定义知 τ_1 和 τ_2 有最小和最大不动点。根据 CTL 语义知 $[[EGp]]$ 是 τ_1 的最大不动点，$[[E(pUq)]]$ 是 τ_2 的最小不动点，即

$$\begin{array}{lcl} [[EGp]] & = & \nu Z.([[p]] \cap ex(Z)) \\ [[E(pUq)]] & = & \mu Z.([[q]] \cup ([[p]] \cap ex(Z))) \end{array}$$

类似地，其它模态算子也可以用不动点刻画。

定义 $ax(A) = \{s \in S \mid \forall s' \in S.((s, s') \in R \to s' \in A)\}$。则 $[[AXp]] = ax([[p]])$。

为方便书写，我们直接将 p, q 看成 S 的子集，时序算子和逻辑联接符看成是 2^S 上的函数。

AFp	$=$	$\mu Z.(p \vee AXZ)$
AGp	$=$	$\nu Z.(p \wedge AXZ)$
EFp	$=$	$\mu Z.(p \vee EXZ)$
EGp	$=$	$\nu Z.(p \wedge EXZ)$
$A(pUq)$	$=$	$\mu Z.(q \vee (p \wedge AXZ))$
$A(pRq)$	$=$	$\nu Z.(q \wedge (p \vee AXZ))$
$E(pUq)$	$=$	$\mu Z.(q \vee (p \wedge EXZ))$
$E(pRq)$	$=$	$\nu Z.(q \wedge (p \vee EXZ))$

Proposition 6.6 给定 *Kripke* 结构 M 和 *CTL* 公式 φ。$M, s \models \varphi$ 当且仅当 $s \in [[\varphi]]$。

Corollary 6.2 给定标号 *Kripke* 结构 M 和 *CTL* 公式 φ。$M \models \varphi$ 当且仅当 $I \in [[\varphi]]$。

§6.2.1 模态 μ 演算

由关于 CTL 的讨论我们知道 CTL 公式可以用时序算子 AX,EX, 命题变量和不动点表示。在这种表示中，命题变量可以出现在 AX, EX, μ, ν 之后。将这个约束去掉，我们可以得到一个表达能力更强的逻辑。称为模态 μ 演算或简称为 μ 演算。 给定一个原子命题集合 AP ，一个变量集合 V 。用 p 表示 AP 中任意命题， X 表示 V 中任意变量。一种简单的 μ- 演算的公式的集合可由以下语法给出。

$$\phi ::= p \mid X \mid \neg\phi \mid \phi \wedge \phi \mid \phi \vee \phi \mid \langle . \rangle \phi \mid [.]\phi \mid \mu X.\phi \mid \nu X.\phi$$

其中 $\mu X.\phi$ 和 $\nu X.\phi$ 的 ϕ 中不受囿的 X 必须在偶数个 $\neg$ 符号的作用范围之下。

带动作的描述: 对前述简单 μ 演算做扩充, 增加动作的描述, 我们可以得到一种能够描述动作的时序关系的时序逻辑。 给定一个原子命题集合 AP ，一个变量集合 V ，和一个动作集合 A 。用 p 表示 AP 中任意命题， X 表示 V 中任意变量， a 表示 A 中任意动作。 μ 演算公式的集合由以下语法给出。

$$\phi ::= p \mid X \mid \neg\phi \mid \phi \wedge \phi \mid \phi \vee \phi \mid \langle a \rangle \phi \mid [a]\phi \mid \mu X.\phi \mid \nu X.\phi$$

其中 $\mu X.\phi$ 和 $\nu X.\phi$ 的 ϕ 中不受囿的 X 必须在偶数个 $\neg$ 符号的作用范围之下。

μ 演算公式的语义： μ 演算公式在双标号迁移系统上解释。设 $M = \langle \Sigma, S, \Delta, I, L \rangle$ 为原子命题集 AP 上的双标号迁移系统， $e : V \to 2^S$ 为变量到 S 子集的赋值。 μ 演算公式的语义如下：

$[[p]]e$	$=$	$\{s \mid p \in L(s)\}$
$[[X]]e$	$=$	$e(X)$
$[[\neg\phi]]e$	$=$	$S \setminus [[\phi]]e$
$[[\phi_1 \wedge \phi_2]]e$	$=$	$[[\phi_1]]e \cap [[\phi_2]]e$
$[[\phi_1 \vee \phi_2]]e$	$=$	$[[\phi_1]]e \cup [[\phi_2]]e$
$[[\langle a \rangle \phi]]e$	$=$	$\{s \mid \exists s'.s \xrightarrow{a} s' \wedge s' \in [[\phi]]e\}$
$[[[a]\phi]]e$	$=$	$\{s \mid \forall s'.s \xrightarrow{a} s' \Rightarrow s' \in [[\phi]]e\}$
$[[\mu X.\phi]]e$	$=$	$\cap\{S' \subseteq S \mid [[\phi]]e[X/S'] \subseteq S'\}$
$[[\nu X.\phi]]e$	$=$	$\cup\{S' \subseteq S \mid S' \subseteq [[\phi]]e[X/S']\}$

所有变量都是受囿变量的公式称为闭公式。闭公式的语义不受 e 的影响。设 M 为双标号迁移系统， φ 为闭公式。 φ 在 M 中的语义可写为 $[[\varphi]]$ 。

Definition 6.3 $M \models \varphi$ 当且仅当 $I \subseteq [[\varphi]]$ 。

可满足公式: 设 φ 为 μ 演算公式。 φ 是可满足的，当且仅当存在 M 使得 $M \models \varphi$ 。

重言式和等价: 设 φ 为 μ 演算公式。 φ 是重言式, 记作 $\models \varphi$ ，当且仅当对任意 M 有 $M \models \varphi$ 。 φ 等价于 ψ ，记作 $\varphi \equiv \psi$ ，当且仅当 $\models (\varphi \wedge \psi) \vee (\neg\varphi \wedge \neg\psi)$ 。

可满足性判定问题: 判定 μ 演算公式是否可满足的问题复杂性为 EXPTIME 完全。

模型检测问题: 判定模型是否满足 μ 演算公式的问题复杂性为 NP $\cap$ coNP 。

§6.2.2 模态算子的对偶关系与 NNF 范式:

逻辑连接符 $\neg$ 只出现在命题前面的公式称为 NNF 范式。每个 μ 演算公式等价于一个 μ 演算的 NNF 范式。一个 μ 演算的 NNF 范式可应用以下对偶关系构造。

$[a]\phi$	$\equiv$	$\neg\langle a\rangle\neg\phi$
$\mu X.\phi$	$\equiv$	$\neg\nu X.\neg\phi(\neg X/X)$

§6.2.3 CTL 公式到 μ 演算公式的转换

由于 CTL 公式不牵涉到动作的描述，可以假定 CTL 所描述系统中，只用同一个动作，记作 a 。这样，可以用以下规则将 CTL 公式转换到 μ 演算公式。

$$\begin{array}{lcl} T(p) & = & p \\ T(\neg\varphi) & = & \neg T(\varphi) \\ T(\varphi\vee\psi) & = & T(\varphi)\vee T(\psi) \\ T(EX\varphi) & = & \langle a\rangle T(\varphi) \\ T(E(\varphi U\psi) & = & \mu Y.(T(\psi)\vee(T(\varphi)\wedge\langle a\rangle Y)) \\ T(EG\varphi) & = & \nu Y.(T(\varphi)\wedge\langle a\rangle Y) \end{array}$$

Proposition 6.7 给定 AP 上的 *Kripke* 模型 $M=\langle S,R,I,L\rangle$ 。定义 $M'=\langle\{a\},S,\Delta,I,L\rangle$ 其中 $\Delta=\{(s,a,s')\mid(s,s')\in R\}$ 。设 φ 为 *CTL* 公式。 $M\models\varphi$ 当且仅当 $M'\models T(\varphi)$ 。

§6.3 计算树逻辑 CTL*

由于 CTL 中描述分枝情况和描述状态的前后关系的算子成对出现，在一定程度上限制了 CTL 的表达能力。我们可以将其拆开使用。这个逻辑记作 CTL* 。给定一个原子命题集合 AP 。用 p 表示 AP 中任意命题。 CTL* 公式的集合由以下语法给出。

$$\phi ::= p\mid\neg\phi\mid\phi\wedge\phi\mid\phi\vee\phi\mid X\,\phi\mid F\,\phi\mid G\,\phi\mid\phi\,U\,\phi\mid\phi\,R\,\phi\mid A\phi\mid E\phi$$

我们可以将 CTL* 中的公式分为状态公式和路径公式。

若 $p\in AP$,	则 p 是状态公式
若 φ 和 ψ 是状态公式,	则 $\neg\varphi,\varphi\vee\psi,\varphi\wedge\psi$ 是状态公式
若 φ 是路径公式,	则 $E\varphi$ 和 $A\varphi$ 是状态公式
若 φ 是状态公式,	则 φ 是路径公式
若 φ 和 ψ 是路径公式,	则 $\neg\varphi,\varphi\vee\psi,\varphi\wedge\psi,X\varphi,F\varphi,G\varphi,\varphi U\psi,\varphi R\psi$ 是路径公式

CTL* 公式在一个树状结构或 Kripke 结构上解释。设 $M=\langle S,R,I,L\rangle$ 是原子命题集 AP 上的 Kripke 结构。 CTL* 公式的语义如下：

$M, s \models p$	若 $p \in AP$ 且 $p \in L(s)$
$M, s \models \neg\varphi$	若 $M, s \not\models \varphi$
$M, s \models \varphi \vee \psi$	若 $M, s \models \varphi$ 或 $M, s \models \psi$
$M, s \models \varphi \wedge \psi$	若 $M, s \models \varphi$ 且 $M, s \models \psi$
$M, s \models A\varphi$	若对于所有 M 中 s 为起点的路径 π ： $M, \pi \models \varphi$
$M, s \models E\varphi$	若存在 M 中 s 为起点的路径 π ： $M, \pi \models \varphi$
$M, \pi \models p$	若 $p \in AP$ 且 $M, \pi_0 \models p$
$M, \pi \models A\varphi$	若 $M, \pi_0 \models A\varphi$
$M, \pi \models E\varphi$	若 $M, \pi_0 \models E\varphi$
$M, \pi \models \neg\varphi$	若 $M, \pi \not\models \varphi$
$M, \pi \models \varphi \vee \psi$	若 $M, \pi \models \varphi$ 或 $M, \pi \models \psi$
$M, \pi \models \varphi \wedge \psi$	若 $M, \pi \models \varphi$ 且 $M, \pi \models \psi$
$M, \pi \models X\varphi$	若 $M, \pi^1 \models \varphi$
$M, \pi \models G\psi$	若 $\forall i \geq 0, M, \pi^i \models \psi$
$M, \pi \models F\varphi$	若 $\exists i \geq 0, M, \pi^i \models \varphi$
$M, \pi \models \varphi U \psi$	若 $\exists i \geq 0, M, \pi^i \models \psi$ 且 $\forall 0 \leq j < i, M, \pi^j \models \varphi$
$M, \pi \models \varphi R \psi$	若 $\forall i \geq k$, 若 $\forall 0 \leq j < i, M, \pi^j \not\models \varphi$ 则 $M, \pi^i \models \psi$

Definition 6.4 *设 φ 为 CTL* 状态公式，M 为 Kripke 结构。$M \models \varphi$ 当且仅当对所有 $s \in I$，$M, s \models \varphi$ 。*

可满足公式: 设 φ 为 CTL* 状态公式。 φ 是可满足的，当且仅当存在 M 使得 $M \models \varphi$ 。

重言式和等价: 设 φ 为 CTL* 状态公式。 φ 是重言式，记作 $\models \varphi$ ，当且仅当对任意 M 有 $M \models \varphi$ 。 φ 等价于 ψ ，记作 $\varphi \equiv \psi$ ，当且仅当 $\models (\varphi \wedge \psi) \vee (\neg\varphi \wedge \neg\psi)$ 。

表达能力: PLTL 和 CTL 是 CTL* 的子集。设 PLTL 、 CTL 和 CTL* 都在 Kripke 结构上解释。一个 PLTL 公式 φ 等价于一个 CTL* 公式 $A\varphi$ 。这样 PLTL 公式就可以看成是状态公式。在这样的解释下，我们可以比较 PLTL 、 CTL 和 CTL* 的状态公式。我们有以下结论。

CTL 公式 $AGEFp$ 不能用 PLTL 表示。
PLTL 公式 $F(p \wedge Xp)$ 不能用 CTL 表示。
CTL* 公式 $E(GFp)$ 不能用 CTL 表示，不能用 PLTL 表示。

Proposition 6.8 *PLTL 和 CTL 互不隶属且 CTL* 大于 PLTL 和 CTL 的并集。*

可满足性判定问题: 判定 CTL* 公式是否可满足的问题复杂性为双指数时间完全。

模型检测问题: 判定模型是否满足 CTL* 公式的问题复杂性为 PSPACE 完全。

§6.4 练习

1. 使用上一章练习 1 的双标号迁移模型 M 。由双标号迁移模型构造标号 Kripke 模型 M' 。用限界语义验证该模型是否满足 $A(q_0 U q_1)$ 和 $EG(q_0 \vee q_1)$ ；分别给出最小的可以确定以上公式是否满足的界。

2. 使用上一题的标号 Kripke 模型 M' 。用不动点算法计算 $[[A(q_0 U q_1)]]$ 和 $[[EG(q_0 \vee q_1)]]$ ；讨论 M' 是否满足 $A(q_0 U q_1)$ 和 $EG(q_0 \vee q_1)$ 。

3. 使用上一章练习 1 的双标号迁移模型 M 。假定投 1 元硬币和投 5 元硬币的动作分别记为 1 和 5 。用不动点算法计算 $\mu Z.(q_1 \vee \langle 1 \rangle Z)$ 和 $\mu Z.(q_1 \vee \langle 5 \rangle Z)$ ；讨论模型是否满足 $\mu Z.(q_1 \vee \langle 1 \rangle Z)$ 和 $\mu Z.(q_1 \vee \langle 5 \rangle Z)$ 。

§7 推理验证方法

本章介绍变量迁移模型是否具备给定性质的推理验证。本章的模型以一阶逻辑为基础。以下假定 $B=(F,P)$ 以及 B 的解释 $I=(D,I_0)$ 为给定。 设 $\varphi,\psi,\phi\in WFF_B$ 为公式。

§7.1 流程图模型的推理

对于流程图模型，传统的性质包括部分正确和完全正确。这些性质约束的是模型运行终止时程序的变量状态，以及模型运行是否能够终止。

设 T 为给定的 (B,V) 上的流程图模型。

定义模型的语义函数为变量状态到变量状态的偏函数 $\mathcal{M}_I(T):\Sigma\hookrightarrow\Sigma$ 如下。

$$\mathcal{M}_I(T)(\sigma)=\begin{cases}\sigma' & \text{若 } (beg,\sigma)\xrightarrow{*}(end,\sigma')\text{ 。}\\ \text{无定义} & \text{若不存在 } \sigma' \text{ 使得 } (beg,\sigma)\xrightarrow{*}(end,\sigma')\text{ 。}\end{cases}$$

$\mathcal{M}_I(T)(\sigma)$ 有定义或者说模型 T 在初始变量状态为 σ 时的运行终止记作 $\mathcal{M}_I(T)(\sigma)\downarrow$ 。反之则记作 $\mathcal{M}_I(T)(\sigma)\uparrow$ 。设 p 和 q 是两个公式。

- 部分正确性： $\models_I\{p\}T\{q\}$ 当且仅当 $\forall\sigma.(I(p)(\sigma)\to(\mathcal{M}_I(T)(\sigma)\downarrow\to I(q)(\mathcal{M}_I(T)(\sigma))))$ ；
- 终止性： $\models_I[p]T[true]$ 当且仅当 $\forall\sigma.(I(p)(\sigma)\to(\mathcal{M}_I(T)(\sigma)\downarrow))$ 。
- 完全正确性： $\models_I[p]T[q]$ 当且仅当 $\forall\sigma.(I(p)(\sigma)\to(\mathcal{M}_I(T)(\sigma)\downarrow\wedge I(q)(\mathcal{M}_I(T)(\sigma))))$ 。

§7.1.1 直接证明

对于部分正确要求而言 (设前置谓词和后置谓词分别为 φ 和 ψ) ，证明模型满足部分正确的一种思路是假设模型的运行为

$$(beg,\sigma_0),(l_1,\sigma_1),...,(l_{n-1},\sigma_{n-1}),(end,\sigma_n)$$

然后证明若 $\varphi(\sigma_0)$ 则 $\psi(\sigma_n)$ 。

对于终止而言 (设前置谓词为 φ) ，一种证明思路是假设模型的运行不终止

$$(beg,\sigma_0),(l_1,\sigma_1),...,$$

那么就会有某个 $l\in LB$ 在运行中出现无限多次。然后证明若 $\varphi(\sigma_0)$ 成立则 l 在某次出现后，与其所对应的 σ 不满足还能够回去执行定义 l 的指令的条件。

完全正确性可以分成部分正确和终止两方面的证明。

如果直接证明完全正确性，那么一种思路是首先假设模型的运行为

$$(beg,\sigma_0),(l_1,\sigma_1),.....$$

然后证明若 $\varphi(\sigma_0)$ 成立则通过 n 次 (n 与初始变量状态 σ_0 相关) 状态变化后，我们有 $l_n=end$ 且 $\psi(\sigma_n)=true$ 。

§7.1.2 基于路径的推理

以上的的证明思路是基于对模型运行过程的分析。对于简单的程序，该思路是可行的。 对于复杂的模型，一个重要的原则是将一个大的问题分解成小的问题。对流程图模型而言，我们可以将模型的运行分成不同的片段，由这些片段的正确性推导整个模型的正确性。 模型片段用标号序列刻画。称标号序列为路径。

Definition 7.1 给定模型 T 。标号序列 $(l_0, l_1, ..., l_n)$ 是 T 的路径当且仅当 $n > 0$ 且对所有 $0 \leq i < n$ ， l_{i+1} 在 l_i 的定义中出现。

设 $\alpha = (l_0, l_1, ..., l_n)$ 为一路径。路径的语义定义如下。

$$\mathcal{M}_I(\alpha)(\sigma_0) = \begin{cases} \sigma_n & \text{若存在 } \{\sigma_1, ..., \sigma_n\} \subseteq \Sigma \\ & \text{使得 } (l_0, \sigma_0) \to (l_1, \sigma_1) \to \cdots \to (l_n, \sigma_n) \text{ 。} \\ \text{无定义} & \text{若不存在这样的状态集。} \end{cases}$$

$\mathcal{M}_I(\alpha)(\sigma)$ 有定义或者说路径 α 在初始变量状态为 σ 时的运行完成记作 $\mathcal{M}_I(\alpha)(\sigma)\downarrow$ 。反之则记作 $\mathcal{M}_I(\alpha)(\sigma)\uparrow$ 。

Proposition 7.1 若 $l_0 = beg$ 且 $l_n = end$ 则 $\alpha = (l_0, l_1, ..., l_n)$ 称为完整路径。 $\mathcal{M}_I(T)(\sigma) = \sigma'$ 当且仅当存在完整路径 α 使得 $\mathcal{M}_I(\alpha)(\sigma) = \sigma'$ 。

类似于模型正确性，我们可以定义路径的部分正确性。

Definition 7.2 设 p 和 q 是两个公式。 α 在 I 解释之下对于 p 和 q 是部分正确的，记作 $\models_I \{p\}\alpha\{q\}$ ，若 $\forall\sigma.(I(p)(\sigma) \to (\mathcal{M}_I(\alpha)(\sigma)\downarrow \to I(q)(\mathcal{M}_I(\alpha)(\sigma))))$ 。

Corollary 7.1 $\models_I \{p\}T\{q\}$ 当且仅当对所有完整路径 α 有 $\models_I \{p\}\alpha\{q\}$ 。

Definition 7.3 设 α 为路径， p 和 q 是两个公式。 p 称为 α 和 q 的最弱宽松前断言，记作 $p = wlp(\alpha, q)$ ，当且仅当 $I(p)(\sigma) \leftrightarrow (\mathcal{M}_I(\alpha)(\sigma)\downarrow \to I(q)(\mathcal{M}_I(\alpha)(\sigma)))$ 。

Definition 7.4 设 $\alpha = (l_0, ..., l_k)$ 为一路径。定义 $[\alpha]q$ 如下。

$k = 1$	存在 T 中指令 l_0: $(v_1, ..., v_n) := (e_1, ..., e_n)$ *goto* l_1 则 $[\alpha]q = q[v_1/e_1]\cdots[v_n/e_n]$.
	存在 T 中指令 l_0: *if* (b) *goto* l *else goto* l' *fi* 则 $[\alpha]q = \begin{cases} b \to q & \text{若 } l = l_1; \\ \neg b \to q & \text{若 } l' = l_1 \text{ 。} \end{cases}$
$k > 1$	设 $\alpha_0 = (l_0, l_1)$, $\alpha_1 = (l_1, ..., l_k)$ 。则 $[\alpha]q = [\alpha_0][\alpha_1]q$

Proposition 7.2 $wlp(\alpha, q) \equiv [\alpha]q$ 。

Corollary 7.2 若 $\models_I p \to [\alpha]q$ 则 $\models_I \{p\}\alpha\{q\}$ 。

$p \to [\alpha]q$ 称为 $\{p\}\alpha\{q\}$ 的验证条件。

Definition 7.5 设 T 为模型， C 为标号集合。定义 $\gamma_T(C)$ 如下：
$(l_0, ..., l_k) \in \gamma_T(C)$ 当且仅当 $(l_0, ..., l_k)$ 是 T 的路径， $l_0, l_k \in C$ 且 $l_1, ..., l_{k-1} \notin C$ 。

部分正确性证明:

Proposition 7.3 设 C 是标号集合， $beg, end \in C$, T 的每个循环至少有一个标号包含于 C 且 C 中的每个标号 l 有一个对应的公式 q_l 。若 $\forall\alpha = (l_0, ..., l_k) \in \gamma_T(C)$, $\models_I p \to [\alpha]q$ ，则 $\models \{q_{beg}\}T\{q_{end}\}$ 。

终止性的特点：终止性不是一阶性质。以下模型在一阶算术 PA (Peano Arithmetic) 的标准模型下对任意输入能够终止，但在 PA 的非标准模型并不一定终止。

beg:	$(x) := (0)$ goto test
test:	if $x = y$ then goto end else goto loop fi
loop:	$(x) := (x + 1)$ goto test

基于谓词的终止性证明：

Proposition 7.4 设 C 是标号集合，$beg \in C$, T 的每个循环至少有一个标号包含于 C 且 C 中的每个标号 l 有一个对应的公式 q_l 。$(W, \sqsubseteq)$ 是一 WFS 。$C' \subseteq C$ 是标号集合，T 的每个循环至少有一个标号包含于 C' 且 C' 中的每个标号 l 有一个对应的函数 $g_l : \Sigma \to W$ 。若以下条件成立则 $\models [q_{beg}]T[true]$ 。

$$
\begin{array}{ll}
(a) & \forall \alpha = (l_0, ..., l_k) \in \gamma_T(C), \models_I vc(q_{l_0}, \alpha, q_{l_k}) \\
(b) & \forall \alpha = (l_0, ..., l_k) \in \gamma_T(C), I(q_{l_0})(\sigma) = true \wedge M_I(\alpha)(\sigma) \downarrow \to g_{l_k}(M_I(\alpha)(\sigma)) \sqsubset q_{l_0}(\sigma)
\end{array}
$$

基于谓词公式的终止性证明：

Proposition 7.5 设 C 是标号集合，$beg \in C$, T 的每个循环至少有一个标号包含于 C 且 C 中的每个标号 l 有一个对应的公式 q_l 。$(W \subseteq D, I_0(\sqsubseteq))$ 是一良基集合且 $\sqsubseteq \in P$ 。w 为最多有一个自由变量的公式且 $W = \{\sigma(x) \mid I(w)(\sigma) = true\}$ 。$C' \subseteq C$ 是标号集合，T 的每个循环至少有一个标号包含于 C' ，C' 中的每个标号 l 有一个对应的项 t_l 且 $\models_I q_l \to w[x/t_l]$ 。若以下条件成立则 $\models [q_{beg}]T[true]$ 。

$$
\begin{array}{ll}
(a) & \forall \alpha = (l_0, ..., l_k) \in \gamma_T(C), \models_I q_{l_0} \to [\alpha] q_{l_k} \\
(b) & \forall \alpha = (l_0, ..., l_k) \in \gamma_T(C), \models_I q_{l_0} \wedge t_{l_0} = a \to [\alpha](t_{l_k} \sqsubset a)
\end{array}
$$

这个方法与上述方法相比有一定的局限性，即良基集合的选择受限于 $W \subseteq D$ 且谓词符号必须在 P 中。这个方法的优点是推理过程简单，不用直接考虑状态的内容。

§7.2 结构化程序模型的推理

流程图模型的一个缺点是循环的入口和出口不规范，从证明的角度讲，推理比较复杂。从模型结构来讲，可组合性较差。结构化程序模型可以克服这些缺点。为方便叙述，亦称结构化程序模型为结构化程序。

设 T 为给定的 (B, V) 上的结构化程序。

定义结构化程序的语义函数为变量状态到变量状态的偏函数 $\mathcal{M}_I(T) : \Sigma \hookrightarrow \Sigma$ 如下。

$$
\mathcal{M}_I(T)(\sigma) = \begin{cases} \sigma' & \text{若 } (T, \sigma) \xrightarrow{*} (\epsilon, \sigma') \text{ 。} \\ \text{无定义} & \text{若不存在 } \sigma' \text{ 使得 } (T, \sigma) \xrightarrow{*} (\epsilon, \sigma') \text{ 。} \end{cases}
$$

$\mathcal{M}_I(T)(\sigma)$ 有定义或者说结构化程序 T 在初始变量状态为 σ 时的运行终止记作 $\mathcal{M}_I(T)(\sigma) \downarrow$。反之则记作 $\mathcal{M}_I(T)(\sigma) \uparrow$ 。设 p 和 q 是两个公式。

- 部分正确性：$\models_I \{p\}T\{q\}$ 当且仅当 $\forall \sigma.(I(p)(\sigma) \to (\mathcal{M}_I(T)(\sigma) \downarrow \to I(q)(\mathcal{M}_I(T)(\sigma))))$ ；
- 终止性：$\models_I [p]T[true]$ 当且仅当 $\forall \sigma.(I(p)(\sigma) \to (\mathcal{M}_I(T)(\sigma) \downarrow))$ 。
- 完全正确性：$\models_I [p]T[q]$ 当且仅当 $\forall \sigma.(I(p)(\sigma) \to (\mathcal{M}_I(T)(\sigma) \downarrow \wedge I(q)(\mathcal{M}_I(T)(\sigma))))$ 。

由于结构化程序可以看成是流程图模型的特殊形式。流程图模型的证明方法同样适用于结构化程序。另外，由于结构化程序具有组合性，我们可以根据指称语义和公理语义证明结构化程序性质。

§7.2.1 指称语义

结构化程序具有组合性。比如给定结构化程序 T_1 和 T_2，我们可将其组合成 $T_1;T_2$，再给一个公式 e，我们可将其组合成 if e then T_1 else T_2 fi 等。这样，$T_1;T_2$ 的语义就可以建立在 T_1 和 T_2 的语义上。比如 $\mathcal{M}_I(T_1;T_2)(\sigma)=\mathcal{M}_I(T_2)\mathcal{M}_I(T_1)(\sigma)$。由于 $\mathcal{M}_I(T_1)(\sigma)$ 不一定有定义，因此以上写法不一定是良定义的。为绕过这个问题，我们对 $\mathcal{M}$ 的定义域和值域进行扩充。定义 $\Sigma_\omega=\Sigma\cup\{\omega\}$。扩充后的语义泛函记为 $\mathcal{M}_I^\varphi$。给定结构化程序 T，$\mathcal{M}_I^\varphi(T)$ 是 Σ_ω 上的函数。

定义 $ite:Bool\times\Sigma_\omega\times\Sigma_\omega\to\Sigma_\omega$ 如下

$$\begin{aligned} ite(true,\sigma,\sigma') &= \sigma \\ ite(false,\sigma,\sigma') &= \sigma' \end{aligned}$$

定义 $\Phi_{e,h}:[\Sigma_\omega\to\Sigma_\omega]\to[\Sigma_\omega\to\Sigma_\omega]$ 如下

$$\Phi_{e,h}(f)(\sigma)=\begin{cases}\omega & \text{若 } \sigma=\omega\\ ite(I(e)(\sigma),f(h(\sigma)),\sigma) & \text{若 } \sigma\neq\omega\end{cases}$$

定义 Σ_ω 上的偏序关系 $\leq$ 如下：$a\leq b$ 当且仅当 $a=\omega$。$(\Sigma_\omega,\leq)$ 是完备偏序。定义 $\mathcal{M}_I^\varphi(T):\Sigma_\omega\to\Sigma_\omega$ 如下。

$\mathcal{M}_I^\varphi(x:=t)(\sigma)$	$=$	$\begin{cases}\omega & \text{若 } \sigma=\omega\\ \mathcal{M}_I(x:=t)(\sigma)=\sigma[x/I(t)(\sigma)] & \text{若 } \sigma\neq\omega\end{cases}$	
$\mathcal{M}_I^\varphi(T_1;T_2)$	$=$	$\mathcal{M}_I^\varphi(T_2)\mathcal{M}_I^\varphi(T_1)$	
$\mathcal{M}_I^\varphi(\text{if } (e) \text{ then } T_1 \text{ else } T_2 \text{ fi})(\sigma)$	$=$	$\begin{cases}\omega & \text{若 } \sigma=\omega\\ ite(I(e)(\sigma),\mathcal{M}_I^\varphi(T_1)(\sigma),\mathcal{M}_I^\varphi(T_2)(\sigma)) & \text{若 } \sigma\neq\omega\end{cases}$	
$\mathcal{M}_I^\varphi(\text{while } e \text{ do } T_1 \text{ od})$	$=$	$\mu\Phi_{e,\mathcal{M}_I^\varphi(T_1)}$	

对于任意结构化程序 T，$\mathcal{M}_I^\varphi(T)$ 是良定义且连续的。$\mathcal{M}_I^\varphi(T)$ 的定义和之前 $\mathcal{M}_I(T)$ 的定义是一致的，即我们有如下结论。

$$\mathcal{M}_I(T)(\sigma)=\begin{cases}M^\omega(T)(\sigma) & \text{若 } M_I^\omega(T)(\sigma)\neq\omega\\ \text{无定义} & \text{若 } M_I^\omega(T)(\sigma)=\omega\end{cases}$$

反过来，我们有

$$\mathcal{M}_I^\varphi(T)(\sigma)=\begin{cases}\omega & \text{若 } \sigma=\omega\text{。}\\ \omega & \text{若 } \mathcal{M}_I(T)(\sigma) \text{ 无定义}\\ \sigma' & \text{若 } \mathcal{M}_I(T)(\sigma)=\sigma'\end{cases}$$

部分正确性：部分正确性的要求证明 $\varphi(\sigma)\to(\mathcal{M}_I(T)(\sigma)\downarrow\to\psi(\mathcal{M}_I(T)(\sigma)))$。如果能够直接通过计算把 $\mathcal{M}_I(T)$ 表示成一个简单函数，则证明就容易了。通常由于部分正确不要求程序运行一定终止，我们只要计算满足 $\mathcal{M}_I^\varphi(T)\sqsubseteq h$ 的一个 Σ 上的函数 h，然后证明 $\varphi(\sigma)\to\psi(h(\sigma))$。

循环语句的特殊处理：应用指称语义进行证明，困难的部分是循环语句的语义。在证明中通常需要定义一个不动点 g 来对应循环语句的语义，以证明循环语句初始时的状态和结束时的状态的关系。我们可以直接定义一个证明循环语句初始时的状态和结束时的状态的关系的方法。

设 $\varphi:\Sigma^2\to Bool$ 为谓词。设 T=while e do T_1 od。若以下命题成立，则 $\forall\sigma\in\Sigma, M_I(T)(\sigma)\downarrow\to\varphi(\sigma,M_I(T)(\sigma))$。

$$\boxed{\begin{array}{l}\forall\sigma\in\Sigma, I(\neg e)(\sigma)\to\varphi(\sigma,\sigma)\\ \forall\sigma,\sigma'\in\Sigma, I(e)(\sigma)\wedge M_I(T_1)(\sigma)\downarrow\wedge\varphi(M_I(T_1)(\sigma),\sigma')\to\varphi(\sigma,\sigma')\end{array}}$$

这里我们只计算 $M_I(T_1)(\sigma)$ 而避开了不动点，因此应用该方法可简化一些循环语句正确性的证明。

终止性: 结构化程序 T 的终止性可以根据已知结构化程序 T' 的终止性来证明。若 $M_I(T')(\sigma)$ 为处处有定义的函数且 $M_I(T)(\sigma)\sqsubseteq M_I(T')(\sigma)$ 则 $M_I(T)(\sigma)$ 处处有定义。

§7.2.2 Hoare 逻辑

除了基于指称语义的证明，我们还可以用逻辑的方法来证明结构化程序的性质。Hoare 逻辑就是这样的一个可以用来证明结构化程序性质的系统。Hoare 逻辑语言建立在谓词逻辑和结构化程序的基础上。

Definition 7.6 *Hoare 公式的定义如下。*

$$H ::= \{p\}T\{q\}$$

其中 $p,q\in WFF_B$ 是公式， $T\in\mathcal{L}_{\bigcirc}^{(B,V)}$ 是结构化程序。

Hoare 公式的语义泛函，记作 I，是基本符号集 B 的解释 I 的扩展。I 将每个 Hoare 公式 $\{p\}T\{q\}$ 映射到一个 $\Sigma\to Bool$ 的函数。$I(\{p\}T\{q\}):\Sigma\to Bool$ 定义如下。

$$\boxed{\begin{array}{c}\mathcal{I}(\{p\}T\{q\})(\sigma)=true\\ \text{当且仅当}\\ \mathcal{I}(p)(\sigma)=true\wedge\mathcal{M}_{\mathcal{I}}(T)(\sigma)\downarrow\to\mathcal{I}(q)(\mathcal{M}_{\mathcal{I}}(T)(\sigma))=true.\end{array}}$$

一个公式 $\{p\}T\{q\}$ 在 I 的解释下成立，即对任意 σ， $\mathcal{I}(\{p\}T\{q\})(\sigma)=true$，记为 $\models_{\mathcal{I}}\{p\}T\{q\}$。若其在任意解释下成立，记为 $\models\{p\}T\{q\}$。若其在满足 W 的解释下成立，记为 $W\models\{p\}T\{q\}$。

推理规则: 设 $x\in V$， $t\in T_B$，设 $e\in QFF_B$， $p,q,r,s\in WFF_B$ 且 $T,T_1,T_2\in\mathcal{L}_{\bigcirc}^{(B,V)}$。我们有以下公理和推理规则。

(1) 赋值公理:

$$\{p[t/x]\}x:=t\{p\}$$

(2) 组合规则:

$$\frac{\{p\}T_1\{r\},\{r\}T_2\{q\}}{\{p\}T_1;T_2\{q\}}$$

(3) 条件规则:

$$\frac{\{p\wedge e\}T_1\{q\},\{p\wedge\neg e\}T_2\{q\}}{\{p\}\text{if } e \text{ then } T_1 \text{ else } T_2 \text{ fi}\{q\}}$$

(4) 循环规则:

$$\frac{\{p \wedge e\}T_1\{p\}}{\{p\}\text{while } e \text{ do } T_1 \text{ od}\{p \wedge \neg e\}}$$

(5) 推论规则：

$$\frac{p \to q, \{q\}T\{r\}, r \to s}{\{p\}T\{s\}}$$

一个 Hoare 公式 $\{p\}T\{q\}$ 可由以上规则和公理得到，记作 $\vdash \{p\}T\{q\}$ 。

由于推导中需要用到 $p \to r$ 等公式。我们还需要一套推导这些公式的方法。

为了专注于与结构化程序直接相关的性质的推理，我们通常将这些成立的式子归于一个理论，在这理论之下，我们可以将这些当成公理来看待。常用的理论包括自然数理论 PA 等。

推理证明: 结构化程序的推理证明在程序的每个语句前后放上合适的断言，应用以上规则证明程序的每个语句对于这些断言都是对的。有些断言是可以从需要证明的前后断言通过简单推导得到的，有些是要自己通过分析程序添加的。

可靠性: 推理系统的可靠指的是推导出的公式确实是成立的，即： 若 $W \vdash \{p\}T\{q\}$ 则 $W \models \{p\}T\{q\}$ 。 给定 I 。记 $th(I)$ 为在 I 的解释下成立的所有公式。我们需要保证

$$\text{若 } th(I) \vdash \{p\}T\{q\} \text{ 则 } \models_I \{p\}T\{q\} \text{ 。}$$

我们有以下性质。

$\models_I \{q_x^t\}x := t\{q\}$
$\models_I \{p\}T_1\{r\}$ 且 $\models_I \{r\}T_2\{q\}$ ，则 $\models_I \{p\}T_1; T_2\{q\}$
$\models_I \{p \wedge e\}T_1\{q\}$ 且 $\models_I \{p \wedge \neg e\}T_2\{q\}$ ，则 $\models_I \{p\}\text{if } e \text{ then } T_1 \text{ else } T_2 \text{ fi}\{q\}$
$\models_I \{p \wedge e\}T_1\{p\}$ ，则 $\models_I \{p\}$ while e do T_1 od$\{p \wedge \neg e\}$
$\models_I \{q\}T\{r\}$ ， $\models_I p \to q$ 且 $\models_I r \to s$ ，则 $\models_I \{p\}T\{q\}$

这些性质对应于前面的一个公理和 4 个推理规则，保证了推理系统的可靠。

相对的完备性: 定义 φ 为 T 和 ψ 的最弱宽松前断言如下：

$$\varphi(\sigma) = true \text{ 当且仅当 } \mathcal{M}_{\mathcal{I}}(T)(\sigma) \downarrow \to \psi(\mathcal{M}_{\mathcal{I}}(T)(\sigma)) = true.$$

我们有以下性质。

$\models_I \{p\}x := t\{q\}$ ，则 $\models_I p \to q_x^t$
$\models_I \{p\}T_1; T_2\{q\}$ 且 $I(r)$ 为 T_2 和 $I(q)$ 的最弱自由前断言， 则 $\models_I \{p\}T_1\{r\}$ 且 $\models_I \{r\}T_2\{q\}$
$\models_I \{p\}\text{if } e \text{ then } T_1 \text{ else } T_2 \text{ fi}\{q\}$ ，则 $\models_I \{p \wedge e\}T_1\{q\}$ 且 $\models_I \{p \wedge \neg e\}T_2\{q\}$
$\models_I \{p\}$ while e do T_1 od$\{q\}$ ，则 $\models_I \{p\}$ if (e) T_1; while e do T_1 od else $x := x$ fi $\{q\}$

对于完备性，我们需要具有足够强表达能力的 I 。 I 的表达能力足够强的含义就是对任意的结构化程序 T 和公式 q ，存在公式 r 使得 $I(r)$ 为 T 和 $I(q)$ 的最弱自由前断言。在此情况下，我们称谓词 $I(r)$ 是可表达的。

给定 I 且假定 I 的表达能力足够强，对 T 做结构归纳，我们有

$$\models_I \{p\}T\{q\} \text{ 则 } th(I) \vdash \{p\}T\{q\}$$

即 Hoare 演算系统具有相对完备性。

导出规则: 为了推理的方便我们可以使用导出规则。导出规则可以看作是基本规则的组合应用。若用已有的公理和规则可以从公式 $s_1, s_2, ..., s_n$ 推导出 s ，则可以产生以下导出规则。

$$\frac{s_1, s_2, ..., s_n}{s}$$

设 $x \in V$ ， $t \in T_B$ ， $e \in QFF_B$ ， $p, q, r, p_0, ..., p_n \in WFF_B$ 且 $T_1, ..., T_n \in \mathcal{L}_{\bigcirc}^{(B,V)}$ 。以下是一些常用的导出规则。这些规则可用以替代原有的赋值公理，组合规则，条件规则和循环规则。

(1')

$$\frac{p \to q_x^t}{\{p\}x := t\{q\}}$$

(2')

$$\frac{\{p_0\}T_1\{p_1\}, \{p_1\}T_2\{p_2\}, ..., \{p_{n-1}\}T_n\{p_n\}}{\{p_0\}T_1; T_2; ...; T_n\{p_n\}}$$

(3')

$$\frac{p \to (p_1 \wedge e) \vee (p_2 \wedge \neg e), \{p_1\}T_1\{q\}, \{p_2\}T_2\{q\}}{\{p\}\text{if } e \text{ then } T_1 \text{ else } T_2 \text{ fi}\{q\}}$$

(4')

$$\frac{p \to r, \{r \wedge e\}T_1\{r\}, (r \wedge \neg e) \to q,}{\{p\}\text{while } e \text{ do } T_1 \text{ od}\{q\}}$$

循环规则的替代规则: 循环规则中的 p 称为循环不变式。应用 Hoare 演算进行证明的难点在于添加合适的循环不变式。由于这是难点，我们可以考虑从不同侧面证明循环语句的正确性。一个替代循环规则的方法可以是从指称语义证明循环语句的方法中导出。指称语义证明循环语句的方法中的谓词 $\varphi(\sigma, \sigma')$ 就是一种不变式。设 T=while e do T_1 od 且 T 以 σ 为初始状态经过 m 次 T_1 的执行后终止。设 $\sigma' = M_I(T_1)^m(\sigma)$ 。考虑 $\varphi(\sigma, \sigma')$ 为前后状态之间的关系，且:

(1) e 不成立时 $\varphi(M_I(T_1)^m(\sigma), M_I(T_1)^m(\sigma))$ 成立。

(2) e 成立时则 $\varphi(M_I(T_1)^{m-1}(\sigma), M_I(T_1)^m(\sigma))$ ， $\varphi(M_I(T_1)^{m-2}(\sigma), M_I(T_1)^m(\sigma))$ ，直至 $\varphi(\sigma, M_I(T_1)^m(\sigma))$ 成立。

我们用公式 r 表示 $\varphi(\sigma, \sigma')$ 。给定 σ, σ' 。 r 和 φ 有以下关系。

$$\varphi(\sigma, \sigma') = I(r)(\sigma[x_1'/\sigma'(x_1)] \cdots [x_n'/\sigma'(x_n)])$$

设 $p, q, r \in WFF_B, e \in QFF, T_1 \in \mathcal{L}_{\bigcirc}^{(B,V)}$ 。则

$$\frac{\neg e \to r_{x_1', ..., x_n'}^{x_1, ..., x_n}, \{\neg r \wedge e\}T_1\{\neg r\}, (p \wedge r) \to q_{x_1, ..., x_n}^{x_1', ..., x_n'},}{\{p\}\text{while } e \text{ do } T_1 \text{ od}\{q\}}$$

规则中 p, q 只用 x, y 这样的变量，而 r 中用 x, y, x', y' 来表示循环语句开始和结束时的状态的关系。

扩展 Hoare 逻辑: Hoare 逻辑只证明结构化程序的部分正确，不证明结构化程序的终止性和完全正确。为了证明终止性和完全正确，有必要对 Hoare 逻辑进行扩展。给定 (B, V) 。

Definition 7.7 *扩展 Hoare 公式的定义如下。*

$$H ::= [p]T[q]$$

其中 $p, q \in WFF_B$ 是公式， $T \in \mathcal{L}_{\bigcirc}^{(B,V)}$ 是结构化程序。

扩展的 Hoare 公式的语义泛函，记作 I ，是基本符号集 B 的解释 I 的扩展。 I 将每个扩展的 Hoare 公式 $[p]T[q]$ 映射到一个 $\Sigma \to Bool$ 的函数。 $I([p]T[q]) : \Sigma \to Bool$ 定义如下。

$$\begin{array}{|c|}\hline \mathcal{I}([p]T[q])(\sigma) = true \\ \text{当且仅当} \\ \mathcal{I}(p)(\sigma) = true \to \mathcal{M}_{\mathcal{I}}(T)(\sigma)\downarrow \wedge \mathcal{I}(q)(\mathcal{M}_{\mathcal{I}}(T)(\sigma)) = true. \\ \hline \end{array}$$

结构化程序的运行是否终止取决于循环语句。为了保证循环语句的终止，我们引入 WFS 集合的应用。首先我们必须保证能够用公式定义需要的良基集合。这样，对 B 和 $I = (D, I_0)$ 有以下要求

(1) B 包含一个二元谓词符号，记作 $\leq$ 。

(2) D 包含一个子集 W 且 $(W, I_0(\leq))$ 为一良基集合。

(3) 存在 $w \in WFF_B$ 且 w 包含不多于一个变量，记为 x ，使得 $W = \{\sigma(x) \mid I(w)(\sigma) = true\}$ 。

循环规则为

$$\frac{(p \wedge e) \to w[x/t], [p \wedge e \wedge t = y]T_1[p \wedge t < y]}{[p]\text{while } e \text{ do } T_1 \text{ od}[p \wedge \neg e]}$$

对于赋值，组合，条件语句和推论规则，依照 Hoare 逻辑推理规则的形式把 $\{p\}T\{q\}$ 替换成 $[p]T[q]$ 即可。

§7.3 卫式迁移模型的推理

本节介绍以卫式迁移模型的一阶线性时序逻辑性质的推理方法。 给定变量集合 V 。设 $M = (T, \Theta)$ 为给定的 (B, V) 上的卫式迁移模型。

Definition 7.8 *M 满足一阶线性时序逻辑公式 φ ，记作 $M \models_I \varphi$ ，当且仅当 M 的行为满足 φ ，即 $[[M]] \models \varphi$ 。*

Proposition 7.6 *φ 是 M 的安全性质，当且仅当 $M \models_I \Box\varphi$ 。*

Proposition 7.7 *φ 是 M 的必达性质，当且仅当 $M \models_I \Diamond\varphi$ 。*

初始条件 Θ ：

Proposition 7.8 *若 $\Theta \to \varphi$ ，则 $M \models_I \varphi$ 。*

用 $\varphi \Rightarrow \psi$ 表示 $M \models_I \Box(\varphi \to \psi)$ 。

Corollary 7.3 *若 $\Theta \to \varphi$ 且 $\varphi \Rightarrow \Box\psi$ ，则 ψ 是 M 的安全性质。*

Corollary 7.4 *若 $\Theta \to \varphi$ 且 $\varphi \Rightarrow \Diamond\psi$ ，则 ψ 是 M 的必达性质。*

时序算子 O ：

Definition 7.9 设 S 为迁移集合。定义 $\sigma \xrightarrow{S} \sigma'$ 当且仅当存在 $t \in S$ 使得 $\sigma \xrightarrow{t} \sigma'$ 。

Definition 7.10 设 S 为迁移集合。定义 $\models_I \{\varphi\}S\{\psi\}$ 当且仅当 $I(\varphi)(\sigma) \to \forall\sigma'.((\sigma \xrightarrow{S} \sigma') \to I(\psi)(\sigma'))$ 。

Definition 7.11 设 t 为 $p \longrightarrow (x_1, ..., x_n) := (e_1, ..., e_n)$ 。定义 $E(t) = p$ 且 $E(S) = \bigvee_{t\in S} E(t)$ 。

定义 $T^+ = T \cup \{\neg E(T) \longrightarrow (x_1) := (x_1)\}$ 。

Proposition 7.9 若 $\models_I \{\varphi\}T^+\{\psi\}$ ，则 $\varphi \Rightarrow O\psi$ 。

Definition 7.12 设 S 为迁移集合。公式 p 称为 (S, ψ) 的最弱宽松前断言，记作 $p = wlp(S, \psi)$ ，当且仅当

$$I(p)(\sigma) \leftrightarrow \forall\sigma'.((\sigma \xrightarrow{S} \sigma') \to I(\psi)(\sigma'))$$ 。

Proposition 7.10 设 S 为迁移集合。 $\models_I \{\varphi\}S\{\psi\}$ 当且仅当 $\varphi \to wlp(S, \psi)$ 。

Corollary 7.5 若 $\varphi \to wlp(T^+, \psi)$ ，则 $\varphi \Rightarrow O\psi$ 。

最弱宽松前断言的计算：定义 $[S]\psi$ 如下：

$$[\{p \longrightarrow (x_1, ..., x_n) := (e_1, ..., e_n)\}]\psi = (p \to \psi[x_1/e_1] \cdots [x_n/e_n])$$
$$[S]\psi = \bigwedge_{t\in S}[\{t\}]\psi$$

Proposition 7.11 设 S 为迁移集合。 $wlp(S, \psi) \equiv [S]\psi$ 。

Corollary 7.6 若 $\varphi \to [T^+]\psi$ ，则 $\varphi \Rightarrow O\psi$ 。

推理规则：综合以上内容，我们有以下与模型相关的推理规则，包括初始状态规则，赋值规则，迁移集合规则，下一时刻规则，推论规则。

- Θ 规则：

$$\frac{\Theta \to \varphi}{\varphi}$$

- 赋值规则：

$$\frac{\varphi \wedge p \to \psi[x_1/e_1] \cdots [x_n/e_n]}{\varphi \to [\{p \longrightarrow (x_1, ..., x_n) := (e_1, ..., e_n)\}]\psi}$$

- 迁移集合规则：

$$\frac{\varphi \to [S_0]\psi \quad \varphi \to [S_1]\psi}{\varphi \to [S_0 \cup S_1]\psi}$$

- O 规则：

$$\frac{\varphi \Rightarrow (\psi \vee E(T)) \quad \varphi \to [T]\psi}{\varphi \Rightarrow O\psi}$$

- 推论规则：

$$\frac{\varphi' \to \varphi \quad \varphi \to [T]\psi \quad \psi \to \psi'}{\varphi' \to [T]\psi'}$$

导出规则: 设 $e \in T_B$ 为项， $\sqsubseteq$ 为 P 中二元谓词符号， $w \in QFF_B$ 为一元谓词公式 (记其变量为 x) 且 $W = (\{\sigma(x) \mid I(w)(\sigma) = true\}, I_0(\sqsubseteq))$ 为良基集合， v 为没有在公式中出现过的变量。结合一阶线性时序逻辑的推理规则和以上与模型相关的推理规则，我们有以下规则。

$$\frac{\varphi \Rightarrow \phi \quad (\phi \wedge \neg\varphi_0) \to [T]\phi \quad \phi \Rightarrow \psi}{\varphi \Rightarrow \varphi_0 R \psi} \qquad \frac{\varphi \Rightarrow \phi \quad \phi \to [T]\phi \quad \phi \Rightarrow \psi}{\varphi \Rightarrow \Box\psi}$$

$$\frac{\varphi \Rightarrow (\psi \vee \phi) \quad \phi \Rightarrow (\varphi_0 \wedge w_x^e \wedge (\psi \vee E(T))) \quad (\phi \wedge e = v) \to [T](\psi \vee (\phi \wedge e \sqsubset v))}{\varphi \Rightarrow \varphi_0 U \psi} \qquad \frac{\varphi \Rightarrow (\psi \vee \phi) \quad \phi \Rightarrow (w_x^e \wedge (\psi \vee E(T))) \quad (\phi \wedge e = v) \to [T](\psi \vee (\phi \wedge e \sqsubset v))}{\varphi \Rightarrow \Diamond\psi}$$

若 $B = (F, P)$ 的表达能力不够，从实际应用考虑，可对 B 进行适当扩展和增加相应解释，以便于定义推理规则前提中使用的 e 和 w 等元素。

§7.4 练习

1. 证明卫式迁移模型推理的导出规则的正确性。
2. 设计应用加减运算计算最大公约数的算法。分别用流程图模型、结构化程序模型和卫式迁移模型写清楚算法的各有关部分。分别用这三种模型的相关推理方法证明算法的部分正确性和终止性。

§8 模型检测方法

模型检测的含义是检查一个模型是否满足给定的性质，特别是使用计算的方法检查一个模型是否满足给定的性质。模型检测主要包含三类方法：一类基于状态的分析，一类基于符号模型的分析，一类基于时序逻辑限界语义的分析。本章介绍时序逻辑 CTL 和 PLTL 的这三类适用于有穷状态系统的模型检测方法。

§8.1 基于状态的分析

基于状态的分析是一类基本的分析方法，通过查看状态及状态之间的关系判定给定的性质是否成立。

§8.1.1 CTL 性质的状态标号算法

考虑由 $\{\vee, \neg, EX, EG, EU\}$ 构成的 CTL 逻辑联接符和时序算子的完全集。

给定 AP 上的 Kripke 结构 $\langle S, R, I, L\rangle$ 和一个 CTL 公式 φ，该标号算法的主要思想是逐步扩充 L 使得 $L(s)$ 包含每个在 s 满足的 φ 的子公式。

一个状态是否满足原子命题由 L 给定。

假设每个状态是否满足 φ 的子公式已经计算过。对于一个状态是否满足 φ 我们有以下情况。

（ 1 ）若 $\varphi = p$ 且 p 是命题，则无需计算。

（ 2 ）若 $\varphi = \neg\varphi_0$，则对每个 s，若 $\varphi_0 \notin L(s)$，则将 φ 加入 $L(s)$。

（ 3 ）若 $\varphi = \varphi_0 \vee \varphi_1$，则对每个 s，若 $\varphi_0 \in L(s)$ 或 $\varphi_1 \in L(s)$，则将 φ_0 加入 $L(s)$。

（ 4 ）若 $\varphi = EX\varphi_0$，则对每个 s，若存在 t 使得 $R(s,t)$ 且 $\varphi_0 \in L(t)$，则将 φ 加入 $L(s)$。

（ 5 ）若 $\varphi = E(\varphi_0 U \varphi_1)$，则

（ 5a ）对每个 s，若 $\varphi_1 \in L(s)$，则将 φ 加入 $L(s)$。

（ 5b ）对每个 s，若存在 t 使得 $R(s,t)$ 且 $\varphi \in L(t)$，且 $\varphi_0 \in L(s)$，则将 φ 加入 $L(s)$。

（ 5c ）重复 (5b) 直至 L 不起变化。

（ 6 ）若 $\varphi = EG\varphi_0$，则

（ 6a ）计算 $S' = \{s \mid \varphi_0 \in L(s)\}$。

（ 6b ）计算 $\langle S, R\rangle$ 的导出子图 $\langle S', R'\rangle$ 的非平凡强连通分图的顶点集合 S''，并对每个 $s \in S''$，将 φ 加入 $L(s)$。

（ 6c ）对每个 $s \in S'$，若存在 t 使得 $R(s,t)$ 且 $\varphi \in L(t)$，则将 φ 加入 $L(s)$。

（ 6d ）重复 (6c) 直至 L 不起变化。

然后我们可以直接根据已经计算出的 L 求得系统的初始状态是否满足 φ。

Proposition 8.1 $M, s \models \varphi$ 当且仅当 $\varphi \in L(s)$。

Corollary 8.1 $M \models \varphi$ 当且仅当 $\varphi \in \bigcap_{s\in I} L(s)$。

§8.1.2 CTL 性质的不动点算法

给定 AP 上的 Kripke 结构 $\langle S, R, I, L\rangle$ 和一个 CTL 公式 φ，该不动点算法的主要思想是对每个 φ 的子公式逐层计算哪些状态满足这些子公式，直至最后知道哪些状态满足 φ。M 满足 φ 当且仅当 I 中的状态都满足 φ。具体算法见 §6.2 节及推论 6.2。

§8.1.3 PLTL 性质的基于自动机空性检测的算法

由于每个 PLTL 公式等价于一个 NNF 范式且 $\Box$ 和 $\Diamond$ 可用 R 和 U 表示，我们只考虑不含 $\Box$ 和 $\Diamond$ 的 NNF 范式。给定 AP 上的 Kripke 结构 $M = \langle S, R, I, L\rangle$ 和一个 PLTL 公式 φ，该算法的主要思想是将 M 与 $\neg\varphi$ 转换到等价的 Büchi 自动机 B_M 和 $B_{\neg\varphi}$，然后检查 $B_M \cap B_{\neg\varphi}$ 是否为空。

PLTL 公式与 Büchi 自动机: 记 $L(\pi) = [L(\pi_i)]_{i\geq 0}$。$\zeta \in (2^{AP})^\omega$ 满足 φ 当且仅当存在 $M = \langle S, R, I, L\rangle$ 和 π 使得 $L(\pi) = \zeta$ 且 $M, \pi \models \varphi$。

记 $[[\varphi]] \subseteq (2^{AP})^\omega$ 为满足 φ 的字符串的集合。

对任意 PLTL 公式 φ，存在一个 Büchi 自动机 $A = \langle 2^{AP}, S, \Delta, I, F\rangle$ 使得 $\mathcal{L}(A) = [[\varphi]]$。

由于 Büchi 自动机可通过泛 Büchi 自动机的转换得到，以下是一个构造语言等价于 $[[\varphi]]$ 的泛 Büchi 自动机 $B_\varphi = \langle 2^{AP}, S, \Delta, I, F\rangle$ 的方法。

1. 首先构造状态 $x = (\{\varphi\}, \{\}, \{\})$，记 $a(x) = \{\epsilon\}$。将其加入状态集 S。

2. 对任意 $x = (b, c, d) \in S$：

若 $b = \emptyset$，构造状态 $y = (d, \{\}, \{\})$。若 $y \in S$，则修改 $a(y) = a(y) \cup \{x\}$，若 $y \notin S$，记 $a(y) = \{x\}$，并将 y 加入状态集 S。

若 $b = b_0 \cup \{\varphi\}$，我们分以下几种情况：

若 $\varphi = \varphi_0 U \varphi_1$，构造状态 $(b_0 \cup \{\varphi_0\}, c \cup \{\varphi\}, d \cup \{\varphi\})$ 和 $(b_0 \cup \{\varphi_1\}, c \cup \{\varphi\}, d)$。

若 $\varphi = \varphi_0 R \varphi_1$，构造状态 $(b_0 \cup \{\varphi_1\}, c \cup \{\varphi\}, d \cup \{\varphi\})$ 和 $(b_0 \cup \{\varphi_0, \varphi_1\}, c \cup \{\varphi\}, d)$。

若 $\varphi = \varphi_0 \vee \varphi_1$，构造状态 $(b_0 \cup \{\varphi_0\}, c \cup \{\varphi\}, d)$ 和 $(b_0 \cup \{\varphi_1\}, c \cup \{\varphi\}, d)$。

若 $\varphi = \varphi_0 \wedge \varphi_1$，构造状态 $(b_0 \cup \{\varphi_0, \varphi_1\}, c \cup \{\varphi\}, d)$。

若 $\varphi = O\varphi_0$，构造状态 $(b_0, c \cup \{\varphi\}, d \cup \{\varphi_0\})$。

若 $\varphi = p$ 或 $\varphi = \neg p$，且 $p \in AP$，构造状态 $(b_0, c \cup \{\varphi\}, d)$。

用新构造的状态替代原状态 x。首先对每个新构造的状态 y，若 $y \notin S$，记 $a(y) = a(x)$，并将 y 加入状态集 S；若 $y \in S$ 且 $a(y) \neq \emptyset$，则修改 $a(y) = a(y) \cup a(x)$; 若 $y \in S$ 且 $a(y) = \emptyset$，则对所有直接或间接替代 y 的状态 $y' \in S$，修改 $a(y') = a(y') \cup a(x)$; 然后将所有 $a(z)$ 中出现 x 的地方用新构造的状态替代；最后修改 $a(x) = \emptyset$ 表示 x 已为新构造的状态所替代。

对每个新构造的状态 $y = (b, c, d) \in S$，若 c 中命题的集合是不可满足的，则从 S 中删除 y；若 $c \cap b \neq \emptyset$，则从 b 中删除公共部分 $c \cap b$。返回第二步直至没有新构造的状态。

3. 对任意 $x = (b, c, d) \in S$，若 $y \in a(x)$ 且 $y = (b', c', d') \in S$，构造迁移 (y, σ, x)，并将其加入迁移集合 Δ，其中 $\sigma \in 2^{AP}$ 满足 $\forall p \in AP.(\{p, \neg p\} \not\subseteq \sigma \cup c') \wedge (p \in c' \rightarrow p \in \sigma)$。

4. 对任意 $x = (b, c, d) \in S$，若 $\epsilon \in a(x)$，则将 x 加入初始状态集合 I。

5. 对每个公式 φ，若 $\varphi = \varphi_0 U \varphi_1$，构造 $f_\varphi = \{(b, c, d) \in S \mid \varphi \in c \rightarrow \varphi_1 \in c\}$，并将其加入接受状态集集合 F。

Proposition 8.2 *给定 PLTL 公式 φ。定义 $B_\varphi = \langle 2^{AP}, S, \Delta, I, F\rangle$ 其中 S, Δ, I, F 的构造方法如上。则 $\mathcal{L}(B_\varphi) = [[\varphi]]$ 且 $|S| \leq 2^{|\varphi|}$。*

Corollary 8.2 *给定 PLTL 公式 φ。φ 可满足当且仅当 $\mathcal{L}(B_\varphi)$ 非空。*

Proposition 8.3 *设 φ 为 PLTL 公式。$M \models \varphi$ 当且仅当 $B_M \cap B_{\neg\varphi} \equiv \emptyset$。*

§8.2 基于符号模型的分析

基于符号模型的分析是将状态集合用命题逻辑公式表示，通过对公式的计算来判定给定的性质是否成立。

§8.2.1 符号模型

若 V 为命题集合。定义 $V' = \{v' \mid v \in V\}$ 。记 $\Phi(V)$ 为命题集合 V 上的命题逻辑公式的集合。 记 φ' 为将 φ 中每个变量 x 用 x' 替换后的公式。

Definition 8.1 给定一个命题集合 AP 。一个 AP 上的符号模型是一个四元组 $\langle V, \rho, \Theta, N\rangle$ 其中 V 为命题变量集合， ρ 为 $V \cup V'$ 上的公式， Θ 为 V 上的公式， $N : AP \to \Phi(V)$ 为 AP 到 V 上公式集合 $\Phi(V)$ 的映射。

设 $V = \{v_1, ..., v_n\}$ 。
记 $v = (v_1, ..., v_n)$ 。若 $s = (a_1, ..., a_n)$ ，则 q_v^a 表示 $q_{v_1,...,v_n}^{a_1,...,a_n}$ 。
记 $q_v^a = 1$ 为 $s \models q$ 。

系统状态: 系统状态由命题变量 $v_1, ..., v_n$ 决定。一个系统状态，即 $v_1, ..., v_n$ 的取值，用 n 元组 $(a_1, ..., a_n)$ 表示。系统状态的集合为 $\{0, 1\}^n$ 。
系统的迁移关系 R_ρ 为 $\{((a_1, ..., a_n), (a'_1, ..., a'_n)) \mid (a_1, ..., a_n, a'_1, ..., a'_n) \models \rho\}$ 。
系统的初始状态集合 I_Θ 为 $\{(a_1, ..., a_n) \mid (a_1, ..., a_n) \models \Theta\}$ 。

路径和计算: 我们用 $s \to s'$ 表示存在从 s 到 s' 的迁移, 即 $(s, s') \in R_\rho$ 。符号模型 $M = \langle V, \rho, \Theta, N\rangle$ 上的一条路径是状态集 $\{0, 1\}^n$ 上的一个满足对任意 $i \geq 0$ 有 $s_i \to s_{i+1}$ 的无穷序列 $[s_i]_{i\geq 0}$ 。符号模型 M 上的一次计算是该符号模型上的第一个状态满足 Θ 的一条路径。

符号模型的标号 Kripke 结构: 给定符号模型 $M = \langle V, \rho, \Theta, N\rangle$ 。
定义 $S = \{0, 1\}^n$ 。
定义 $R = \{(s, s') \mid \rho_{v,v'}^{s,s'} = true\}$ 。
定义 $I = \{s \mid \Theta_v^s = true\}$ 。
定义 $L(s) = \{x \in V \mid s \models x\}$ 。
则我们有与 M 对应的 V 上的 Kripke 结构 $K_M = \langle S, R, I, L\rangle$ 。

Definition 8.2 设 φ 为 V 上的 CTL 或 $PLTL$ 公式。 $M \models \varphi$ 当且仅当 $K_M \models \varphi$ 。

标号 Kripke 结构的符号模型: 给定 AP 上的标号 Kripke 结构 $M = \langle S, R, I, L\rangle$ 。我们可以将 S 编号。若 S 有 $\leq 2^n$ 个元素，则可以用 n 个命题 $v_1, ..., v_n$ 来表示。 S 中的第 i 个元素记为 $s(i-1)$ 。
设 $f_0 : S \to \{0, 1\}^n$ 为单射函数。
设 $f_0(s) = (a_1, ..., a_n)$ 。
记 $f_0(s, 0) = \{v_i \mid a_i = 0, i = 1, ..., n\}$ 和 $f_0(s, 1) = \{v_i \mid a_i = 1, i = 1, ..., n\}$ 。
定义 $f_1(s) = \bigwedge_{v\in f_0(s,1)} v \wedge \bigwedge_{v\in f_0(s,0)} \neg v$ 。
定义 $\Theta = \bigvee_{s\in I} f_1(s)$ 。
定义 $f_2(s, s') = f_1(s) \wedge (f_1(s'))'$ 。
定义 $\rho = \bigvee_{(s,s')\in R} f_2(s, s')$ 。
对每个 $p \in AP$ ，定义 $N(p) = \bigvee_{p\in L(s)} f_1(s)$ 。
定义 $V = \{v_1, ..., v_n\}$ 。
则我们有与 $M = \langle S, R, I, L\rangle$ 对应的符号模型 $(M)^{sm} = \langle V, \rho, \Theta, N\rangle$ 。

Proposition 8.4 给定标号 $Kripke$ 模型 M 和 CTL 公式 φ 。 $M \models \varphi$ 当且仅当 $(M)^{sm} \models \varphi$ 。

§8.2.2 CTL 性质的符号模型检测

考虑由 $\{\vee, \neg, EX, EG, EU\}$ 构成的 CTL 逻辑联接符和时序算子的完全集。

给定 $M = (V, \rho, \Theta, N)$ 。 定义 $\exists x.\varphi = (\varphi_x^0 \vee \varphi_x^1)$ 。定义函数 $ex : \Phi(V) \to \Phi(V)$ 如下：

$$ex(\varphi) = \exists x_1' ... x_n'.(\rho(x_1, ..., x_n, x_1', ..., x_n') \wedge \varphi_{x_1, ..., x_n}^{x_1', ..., x_n'})$$

公式的不动点：设 $f : \Phi(V) \to \Phi(V)$ 。定义 $f^0(\varphi) = \varphi$ 和 $f^{i+1}(\varphi) = f^i(\varphi)$ 。 $f(Z)$ 的最小不动点，记作 $\mu Z.f(Z)$，定义为 $f^k(false)$，其中 k 为最小的满足 $f^{i+1}(false) \equiv f^i(false)$ 的 i 。$f(Z)$ 的最大不动点，记作 $\nu Z.f(Z)$ ，定义为 $f^k(true)$ ，其中 k 为最小的满足 $f^{i+1}(true) \equiv f^i(true)$ 的 i 。

CTL 公式的不动点：设 q 是 CTL 公式。我们分 6 种情况计算 q 成立的状态的集合对应的命题逻辑公式。

(1) $q = p_i$ 是原子命题，	则 $[[q]] = N(p_i)$
(2) q 是 $\neg q_0$ ，	则 $[[q]] = \neg[[q_0]]$
(3) q 是 $q_0 \vee q_1$ ，	则 $[[q]] = [[q_0]] \vee [[q_1]]$
(4) q 是 EXq_0 ，	则 $[[q]] = ex([[q_0]])$
(5) q 是 EGq_0 ，	则 $[[q]] = \nu Z.([[q_0]] \wedge ex(Z))$
(6) q 是 $E(q_0 U q_1)$ ，	则 $[[q]] = \mu Z.([[q_1]] \vee ([[q_0]] \wedge ex(Z)))$

Definition 8.3 $M \models \varphi$ 当且仅当 $\Theta \to [[\varphi]]$ 。

Corollary 8.3 给定标号 *Kripke* 模型 M 和 *CTL* 公式 φ 。设 $(M)^{sm} = \langle V, \rho, \Theta, N\rangle$ 。 $M \models \varphi$ 当且仅当 $\Theta \to [[\varphi]]_\rho$ 。

§8.2.3 PLTL 性质的符号模型检测

Definition 8.4 给定一个命题集合 AP 。一个 AP 上的公平符号模型是一个五元组 $\langle V, \rho, \Theta, N, F\rangle$ 其中 $\langle V, \rho, \Theta, N\rangle$ 为符号模型， F 为 V 上的公式的集合。

给定 $M = \langle V, \rho, \Theta, N, F\rangle$ 。

系统状态即是模型 $M' = \langle V, \rho, \Theta, N\rangle$ 的状态。系统的路径和计算即是 M' 的路径和计算。

公平路径和公平计算：无穷序列 $\pi = [s_i]_{i\geq 0}$ 是系统的公平路径，当且仅当 π 是对每一个 $f \in F$ 有无穷多 s_i 满足 f 的路径。无穷序列 $\pi = [s_i]_{i\geq 0}$ 是系统的计算，当且仅当 s_0 满足 Θ 且 π 是系统的公平路径。

公平状态：状态 s 是公平的，当且仅当有一条起点是 s 的公平路径。

Proposition 8.5 定义 $fair = \nu Z. \bigwedge_{f\in F} \mu Y.((f \wedge ex(Z)) \vee ex(Y))$ 。 s 是公平状态当且仅当 s 满足 $fair$ 。

公平符号模型的交：给定 AP 和 AP 上的公平符号模型 $M_1 = \langle V_1, \rho_1, \Theta_1, N_1, F_1\rangle$ 和 $M_2 = \langle V_2, \rho_2, \Theta_2, N_2, F_2\rangle$ 且 $V_1 \cap V_2 = \emptyset$ 。

对每个 $p \in AP$ ，定义 $N(p) = N_1(p) \wedge N_2(p)$ 。

定义 $\rho = \rho_1 \wedge \rho_2 \wedge \bigwedge_{p\in AP}(N_1(p)' \leftrightarrow N_2(p)')$ 且 $\Theta = \Theta_1 \wedge \Theta_2 \wedge \bigwedge_{p\in AP}(N_1(p) \leftrightarrow N_2(p))$ 。

则 $M_1 \cap M_2 = \langle V_1 \cup V_2, \rho, \Theta, N, F_1 \cup F_2\rangle$ 是公平符号模型。

符号模型与公平符号模型的交: 若 $M_1 = \langle V_1, \rho_1, \Theta_1, N_1 \rangle$ 为符号模型且 $M_2 = \langle V_2, \rho_2, \Theta_2, N_2, F_2 \rangle$ 为公平符号模型。则 $M_1 \cap M_2 = (\langle V_1, \rho_1, \Theta_1, N_1, \{\} \rangle \cap \langle V_2, \rho_2, \Theta_2, N_2, F_2 \rangle)$ 。

Proposition 8.6 *若 π 是 M_1 的公平计算且 π' 是 M_2 的公平计算且 $\forall p \in AP. \forall i \geq 0. (\pi_i \models p \leftrightarrow \pi'_i \models p)$ ，则存在 $M_1 \cap M_2$ 的公平计算 π'' 使得 $\forall p \in AP. \forall i \geq 0. (\pi_i \models p \leftrightarrow \pi''_i \models p)$ ；若 π 是 $M_1 \cap M_2$ 的公平计算，则存在 M_1 的公平计算 π' 和且 M_2 的公平计算 π'' 使得 $\forall p \in AP. \forall i \geq 0. ((\pi_i \models p \leftrightarrow \pi'_i \models p) \wedge (\pi_i \models p \leftrightarrow \pi''_i \models p))$ 。*

由 PLTL 公式生成公平符号模型: 给定 PLTL 公式 φ 。

根据 §8.1.3 节的构造生成 S, I, F 。

根据 §8.2.2 节的原理定义 $V = \{v_1, ..., v_n\}$ 以及 f_0, f_1, f_2 。

定义 $\Theta = \bigvee_{s \in I} f_1(s)$ 。

定义 $\rho = \bigvee_{s \in s'.a} f_2((s, s'))$ 。

对每个 $p \in AP$ ，定义 $N(p) = \bigvee_{s.c \models p} f_1(s)$ 。

定义 $F' = \bigvee_{s \in F} f_1(s)$ 。

则 $(\varphi)^{fsm} = \langle V, \rho, \Theta, N, F' \rangle$ 是一个公平符号模型。

Proposition 8.7 *给定标号 Kripke 结构 M 和 PLTL 公式 φ 。设 $(M)^{sm} \cap (\neg\varphi)^{fsm} = \langle V, \rho, \Theta, N, F \rangle$ 。则 $M \models \varphi$ 当且仅当 $\Theta \wedge fair$ 不可满足。*

§8.3 限界模型检测与限界正确性检查

限界模型检测与限界正确性检查算法的实现主要是结合时序逻辑公式的限界语义与符号模型，把模型检测问题转换为逻辑公式的可满足性问题。

限界正确性检查的主要思想是对给定性质，在给定界的限制下，试图证明性质的正确且同时试图证明性质的不正确，因而若两者之一有肯定的答案，则知道模型是否满足性质。限界模型检测的主要思想是对给定性质，将其取反，在给定界的限制下，试图证明模型有初始状态满足取反后的性质，由此得出模型不满足给定性质。

§8.3.1 CTL 公式的限界正确性检查

给定符号模型 $M = \langle V, \rho, \Theta, N \rangle$ 。给定 CTL 公式 φ 。

设 $v = \{v_1, ..., v_n\}$ 。则一个状态变量 w 为一个 n 元组 $(w_1, ..., w_n)$ 。

设 $k \geq 0$ 。

用 $\vec{u}$ 表示状态变量列表 $u_0, ..., u_k$ 。定义 $P_k(\vec{u})$ 如下。

$$P_k(\vec{u}) := \bigwedge_{j=0}^{k-1} T(u_j, u_{j+1})$$

定义 $\gamma_k(\vec{u})$ 如下。

$$\gamma_k(\vec{u}) := \bigvee_{x=0}^{k-1} \bigvee_{y=x+1}^{k} u_x = u_y.$$

Definition 8.5 (CTL 公式的转换) *设 $k \geq 0$. 设 w 为状态变量。定义 $[[\varphi, w]]_k$ 如下。*

$[[p,w]]_k$	$=$	$N(p)_v^w$
$[[\neg p,w]]_k$	$=$	$\neg N(p)_v^w$
$[[\varphi\vee\psi,w]]_k$	$=$	$[[\varphi,w]]_k\vee[[\psi,w]]_k$
$[[\varphi\wedge\psi,w]]_k$	$=$	$[[\varphi,w]]_k\wedge[[\psi,w]]_k$
$[[A\varphi,w]]_k$	$=$	$\forall\vec{u}.(P(\vec{u})\wedge w=u_0\rightarrow[[\varphi,\vec{u}]]_k)$
$[[E\varphi,w]]_k$	$=$	$\exists\vec{u}.(P(\vec{u})\wedge w=u_0\wedge[[\varphi,\vec{u}]]_k)$
$[[X\varphi,\vec{u}]]_k$	$=$	$k\geq 1\wedge[[\varphi,u_1]]_k$
$[[F\psi,\vec{u}]]_k$	$=$	$\bigvee_{j=0}^{k}[[\psi,u_j]]_k$
$[[G\psi,\vec{u}]]_k$	$=$	$\bigwedge_{j=0}^{k}[[\psi,u_j]]_k\wedge\gamma_k(\vec{u}))$
$[[\varphi U\psi,\vec{u}]]_k$	$=$	$\bigvee_{j=0}^{k}([[\psi,u_j]]_k\wedge\bigwedge_{t=0}^{j-1}[[\varphi,u_t]]_k)$
$[[\varphi R\psi,\vec{u}]]_k$	$=$	$\bigwedge_{j=0}^{k}([[\psi,u_j]]_k\vee\bigvee_{t=0}^{j-1}[[\varphi,u_t]]_k)\wedge(\bigvee_{t=0}^{k}[[\varphi,u_t]]_k\vee\gamma_k(\vec{u}))$

Theorem 8.1 给定标号 *Kripke* 模型 M 和 *CTL* 公式 φ。设 $(M)^{sm}=\langle V,\rho,\Theta,N\}$。$M,s\models_k\varphi$ 当且仅当 $f_1(s)\rightarrow[[\varphi,v]]_k$ 成立.

Corollary 8.4 给定标号 *Kripke* 模型 M 和 *CTL* 公式 φ。设 $(M)^{sm}=\langle V,\rho,\Theta,N\}$。$M\models_k\varphi$ 当且仅当 $\Theta\rightarrow[[\varphi,v]]_k$ 成立.

限界正确性检查算法: 给定模型 M 和公式 φ。限界正确性检查算法 $CTLbmc(M,\varphi)$ 如下：

(1) $k=0$；
(2) 若 $\Theta\rightarrow[[\varphi,v]]_k$ 成立，则 return true；
(3) 若 $\exists\vec{v}.(\Theta\wedge[[\neg\varphi,v]]_k)$ 成立，则 return false；
(4) $k=k+1$，回到 (2)；

注意：这里的 $\neg\varphi$ 代表与其等价的 CTL 公式的 NNF 范式。

Corollary 8.5 $M\models\varphi$ *iff* $CTLbmc(M,\varphi)=true$。

§8.3.2 PLTL 公式的限界模型检测

给定符号模型 $M=\langle V,\rho,\Theta,N\rangle$。给定 PLTL 公式 φ。
设 $v=\{v_1,...,v_n\}$。则一个状态变量 w 为一个 n 元组 $(w_1,...,w_n)$。
设 $k\geq 0$。
定义 $s(i,k,l)$ 如下。

$$\text{if } (k=i) \text{ then } l \text{ else } i+1.$$

Definition 8.6 (PlTL 公式的转换) 设 $k\geq 0$. 设 w 为状态变量。用 $[[\varphi,u_{-1}]]_{k,l}$ 表示 false。定义 $[[\varphi,w]]_{k,l}$ 如下。

$[[p,w]]_{k,l}$	$=$	$N(p)_v^w$
$[[\neg p,w]]_{k,l}$	$=$	$\neg N(p)_v^w$
$[[\varphi\vee\psi,w]]_{k,l}$	$=$	$[[\varphi,w]]_{k,l}\vee[[\psi,w]]_{k,l}$
$[[\varphi\wedge\psi,w]]_{k,l}$	$=$	$[[\varphi,w]]_{k,l}\wedge[[\psi,w]]_{k,l}$
$[[X\varphi,u_i]]_{k,l}$	$=$	$[[\varphi,u_{s(i,k,l)}]]_{k,l}$
$[[G\varphi,u_i]]_{k,l}$	$=$	$\bigwedge_{j=i}^{k}[[\varphi,u_j]]_{k,l}\wedge\bigwedge_{j=\min(i,l)}^{i-1}[[\varphi,u_j]]_{k,l}$
$[[\varphi U\psi,u_i]]_{k,l}$	$=$	$\bigvee_{j=i}^{k}([[\psi,u_j]]_{k,l}\wedge\bigwedge_{t=i}^{j-1}[[\varphi,u_t]]_{k,l})\vee$ $\bigwedge_{t=i}^{k}[[\varphi,u_t]]_{k,l}\wedge\bigvee_{j=l}^{i-1}([[\psi,u_j]]_{k,l}\wedge\bigwedge_{t=l}^{j-1}[[\varphi,u_t]]_{k,l})$

Definition 8.7 $[[\varphi, \overrightarrow{u}]]_k := \bigvee_{l=-1}^{k}(T(u_k, u_l) \wedge [[\varphi, u_0]]_{k,l})$

Theorem 8.2 给定标号 *Kripke* 模型 M 和 *PLTL* 公式 φ。设 $(M)^{sm} = \langle V, \rho, \Theta, N\}$。$M, s \models_k^E \varphi$ 当且仅当 $(f_1(s))_v^{u_0} \wedge P_k(\overrightarrow{u}) \wedge [[\varphi, \overrightarrow{u}]]_k$ 可满足.

Corollary 8.6 给定标号 *Kripke* 模型 M 和 *PLTL* 公式 φ。给定任意 $k \geq 0$。若 $\Theta_v^{u_0} \wedge P_k(\overrightarrow{u}) \wedge [[\neg\varphi, \overrightarrow{u}]]_k$ 可满足，则 $M \not\models \varphi$。

Corollary 8.7 给定标号 *Kripke* 模型 M 和 *PLTL* 公式 φ。给定任意 $k \geq 0$。若 $\Theta_v^{u_0} \wedge P_k(\overrightarrow{u}) \wedge [[\neg\varphi, \overrightarrow{u}]]_k$ 不可满足且 $k > ct((M)^{sm}, \varphi)$，则 $M \models \varphi$。

限界模型检测算法: 给定标号 Kripke 模型 M 和 PLTL 公式 φ。设 $(M)^{sm} = \langle V, \rho, \Theta, N\}$。给定参数 b。限界模型检测算法 $PLTLbmc(M, \varphi, b)$ 如下：

(1) $k = 0$
(2) 若 $\Theta_v^{u_0} \wedge P_k(\overrightarrow{u}) \wedge [[\neg\varphi, \overrightarrow{u}]]_k$ 可满足，则 return false；
(3) 若 $k = b$，则 return unknown
(4) $k = k + 1$，回到 (2)

注意：这里的 $\neg\varphi$ 代表与其等价的 PLTL 公式的 NNF 范式。

Corollary 8.8 给定任意 b。若 $PLTLbmc(M, \varphi, b) = false$，则 $M \not\models \varphi$。

Corollary 8.9 给定任意 b 且 $b \geq ct((M)^{sm}, \varphi)$。若 $PLTLbmc(M, \varphi, b) \neq false$，则 $M \models \varphi$。

§8.4 练习

1. 定义由 PLTL 公式构造与其等价的公平符号模型的方法。
2. 设计具有 3 个进程的满足先申请者优先的有穷状态的互斥算法模型。分别用 CTL 和 PLTL 描述需要满足的性质。说明怎样用不同的模型检测方法证明设计的正确性。

§9 实例分析

本章介绍实例。主要实例为互斥算法与整数平方根算法。

§9.1 互斥算法

考虑两个进程的互斥算法。

互斥算法的卫式迁移模型: 给定 $B = (\{s_0, s_1, s_2, s_3, t_0, t_1, t_2, t_3, 0, 1\}, \{=\})$ 。 给定 $I = (D, I_0)$ 其中 $D = \{0, 1, 2, 3\}$ 且 I_0 定义如下:

$I_0(=) =$	$=$
$I_0(s_0) = I_0(t_0) = I(0) =$	0
$I_0(s_1) = I_0(t_1) = I(1) =$	1
$I_0(s_2) = I_0(t_2) =$	2
$I_0(s_3) = I_0(t_3) =$	3

给定 $V = \{a, b, x, y, t\}$ 。

(B, V) 上的 互斥算法的卫式迁移模型定义为 $M = \langle T, \Theta \rangle$ 其中 Θ 为 $a = s_0 \wedge b = t_0 \wedge x = 0 \wedge y = 0$ 且 T 为以下迁移的集合:

$a = s_0$	$\longrightarrow$	$(y, t, a) := (1, 1, s_1)$
$a = s_1 \wedge (x = 0 \vee t = 0)$	$\longrightarrow$	$(a) := (s_2)$
$a = s_2$	$\longrightarrow$	$(y, a) := (0, s_3)$
$a = s_3$	$\longrightarrow$	$(y, t, a) := (1, 1, s_1)$
$b = t_0$	$\longrightarrow$	$(x, t, b) := (1, 0, t_1)$
$b = t_1 \wedge (y = 0 \vee t = 1)$	$\longrightarrow$	$(b) := (t_2)$
$b = t_2$	$\longrightarrow$	$(x, b) := (0, t_3)$
$b = t_3$	$\longrightarrow$	$(x, t, b) := (1, 0, t_1)$

$a = s_i$ 表示进程 A 处于 s_i 状态。 s_0, s_1, s_2, s_3 分别理解为进程 A 的初始状态，申请状态，使用资源状态，普通状态。 类似地， $b = t_i$ 表示进程 B 处于 t_i 状态。

互斥算法的性质: 我们考虑以下三个性质。

$\varphi_1 = \Box(\neg(a = s_2 \wedge b = t_2))$ 。
$\varphi_2 = \Box(a = s_1 \wedge \neg(b = t_1 \vee b = t_2) \rightarrow (a = s_2 \, R \, b \neq t_2))$ 。
$\varphi_3 = \Box(a = s_1 \rightarrow \Diamond a = s_2)$ 。

这三个 PLTL 性质亦可用 CTL 表示如下。

$\varphi_1' = AG(\neg(a = s_2 \wedge b = t_2))$ 。
$\varphi_2' = AG(a = s_1 \wedge \neg(b = t_1 \vee b = t_2) \rightarrow A(a = s_2 \, R \, b \neq t_2))$ 。
$\varphi_3' = AG(a = s_1 \rightarrow AFa = s_2)$ 。

§9.1.1 推理验证

证明: $M \models_I \varphi_1$ 。

应用 §7.3 节的证明结论形式为 $\varphi \Rightarrow \Box\psi$ 的证明规则， $M \models_I \varphi_1$ 可根据以下步骤证明。

首先证明 $M \models_I \Box(\Theta \rightarrow \varphi_1)$ 如下。

定义 $\varphi = \Theta$ 且 $\psi = (\neg(a = s_2 \wedge b = t_2))$ 。

定义 ϕ 为以下公式的析取。

$$
\begin{array}{l}
(a = s_0 \vee a = s_3) \wedge (b = t_0 \vee b = t_3) \\
(a = s_0 \vee a = s_3) \wedge (b = t_1 \vee b = t_2) \wedge x = 1 \\
(b = t_0 \vee b = t_3) \wedge (a = s_1 \vee a = s_2) \wedge y = 1 \\
(a = s_1 \wedge b = t_1 \wedge x = 1 \wedge y = 1) \\
(a = s_1 \wedge b = t_2 \wedge x = 1 \wedge y = 1 \wedge t = 1) \\
(a = s_2 \wedge b = t_1 \wedge x = 1 \wedge y = 1 \wedge t = 0)
\end{array}
$$

则有 $\varphi \Rightarrow \phi$ 且 $\phi \rightarrow [T]\phi$ 且 $\phi \Rightarrow \psi$ 。

因而根据证明规则我们有 $\varphi \Rightarrow \Box\psi$ ，即 $M \models_I \Box(\Theta \rightarrow \varphi_1)$ 。

因而 $M \models_I \Theta \rightarrow \varphi_1$ 。进而我们有 $M \models_I \varphi_1$ 。

证明: $M \models_I \varphi_2$ 。

应用 §7.3 节的证明结论形式为 $\varphi \Rightarrow \varphi_0 R \psi$ 的证明规则， $M \models_I \varphi_2$ 可根据以下步骤证明。

定义 $\varphi, \varphi_0, \psi, \phi$ 如下。

$$
\begin{array}{lcl}
\varphi & = & (a = s_1 \wedge \neg(b = t_1 \vee b = t_2)) \\
\varphi_0 & = & (a = s_2) \\
\psi & = & (b \neq t_2) \\
\phi & = & (a = s_1 \wedge (b = t_0 \vee b = t_3 \vee (b = t_1 \wedge y = 1 \wedge t = 0))) \vee (a = s_2 \wedge b \neq t_2)
\end{array}
$$

则 $\varphi \Rightarrow \phi$ 且 $(\phi \wedge \neg\varphi_0) \rightarrow [T]\phi$ 且 $\phi \Rightarrow \psi$ 。

因而根据证明规则我们有 $\varphi \Rightarrow \varphi_0 R \psi$ ，即 $M \models_I \varphi_2$ 。

证明: $M \models_I \varphi_3$ 。

应用 §7.3 节的结论形式为 $\varphi \Rightarrow \Diamond\psi$ 的证明规则， $M \models_I \varphi_3$ 可根据以下步骤证明。

扩展 B 使得 $B' = (F \cup \{f\}, P \cup \{\leq\})$ 。扩展 I_0 使得以下成立且 $\leq$ 解释为通常的数值大小的比较。

$$
\begin{array}{llll}
I(f(t_0, 0)) = 1 & I(f(t_1, 0)) = 0 & I(f(t_2, 0)) = 2 & I(f(t_3, 0)) = 1 \\
I(f(t_0, 1)) = 1 & I(f(t_1, 1)) = 3 & I(f(t_2, 1)) = 2 & I(f(t_3, 1)) = 1
\end{array}
$$

定义 $\varphi, \psi, W, w, e, \phi$ 如下。

$$
\begin{array}{lcl}
\varphi & = & (a = s_1) \\
\psi & = & (a = s_2) \\
W & = & (\{0, 1, 2, 3\}, \leq) \\
w & = & (0 \leq x \leq 3) \\
e & = & f(b, t) \\
\phi & = & (a = s_1 \wedge y = 1)
\end{array}
$$

则 $\varphi \Rightarrow (\psi \vee \phi)$ 且 $\phi \Rightarrow w_x^e \wedge (\psi \vee E(T))$ 且 $\phi \wedge e = v \rightarrow [T](\psi \vee (\phi \wedge e < v))$ 。

因而根据证明规则我们有 $\varphi \Rightarrow \Diamond\psi$ ，即 $M \models_I \varphi_3$ 。

§9.1.2 基于模型检测算法的验证

假定我们依然验证 φ_1 和 φ_2 这两个性质。那么我们关心的命题的集合为 $\{a = s_1, a = s_2, b = t_1, b = t_2\}$ 。分别用 p_1, p_2, q_1, q_2 表示这些命题。定义 $AP = \{p_1, p_2, q_1, q_2\}$ 。

定义 $a(i) = (i/32)$ ， $b(i) = (i/8)\%4$ ， $x(i) = (i/4)\%2$ ， $y(i) = (i/2)\%2$ ， $t(i) = i\%2$ 。

定义 M 的标号 Kripke 模型 $(M)^{km} = \langle S, R, I, L\}$ 的各分量如下。

- $S = \{w_i \mid 0 \le i \le 127\}$
- $R = R_1 \cup \cdots \cup R_8$ 其中 $R_1, ..., R_8$ 定义如下：

$$
\begin{aligned}
R_1 &= \{(w_i, w_j) \mid a(i) = 0, y(j) = 1, t(j) = 1, a(j) = 1\}\\
R_2 &= \{(w_i, w_j) \mid a(i) = 1, x(i) = 0 \vee t(i) = 0, a(j) = 2\}\\
R_3 &= \{(w_i, w_j) \mid a(i) = 2, y(j) = 0, a(j) = 3\}\\
R_4 &= \{(w_i, w_j) \mid a(i) = 3, y(j) = 1, t(j) = 1, a(j) = 1\}\\
R_5 &= \{(w_i, w_j) \mid b(i) = 0, x(j) = 1, t(j) = 0, b(j) = 1\}\\
R_6 &= \{(w_i, w_j) \mid b(i) = 1, y(i) = 0 \vee t(i) = 1, b(j) = 2\}\\
R_7 &= \{(w_i, w_j) \mid b(i) = 2, x(j) = 0, b(j) = 3\}\\
R_8 &= \{(w_i, w_j) \mid b(i) = 3, x(j) = 1, t(j) = 0, b(j) = 1\}
\end{aligned}
$$

- $I = \{w_0, w_1\}$
- L 定义如下。

  对任意 $i \in \{0, ..., 127\}$ ，对任意 $k \in \{1, 2\}$ ， $p_k \in L(w_i)$ 当且仅当 $a(i) = k$ 。

  对任意 $i \in \{0, ..., 127\}$ ，对任意 $k \in \{1, 2\}$ ， $q_k \in L(w_i)$ 当且仅当 $b(i) = k$ 。

我们有以下结论。

$M \models_I \varphi_1$ 当且仅当 $(M)^{km} \models \Box(\neg(p_2 \wedge q_2))$
$M \models_I \varphi_2$ 当且仅当 $(M)^{km} \models \Box(p_1 \wedge \neg(q_1 \vee q_2) \rightarrow (p_2 \, R \, \neg q_2))$
$M \models_I \varphi_3$ 当且仅当 $(M)^{km} \models \Box(p_1 \rightarrow \Diamond p_2)$
$M \models_I \varphi'_1$ 当且仅当 $(M)^{km} \models AG(\neg(p_2 \wedge q_2))$
$M \models_I \varphi'_2$ 当且仅当 $(M)^{km} \models AG(p_1 \wedge \neg(q_1 \vee q_2) \rightarrow A(p_2 \, R \, \neg q_2))$
$M \models_I \varphi'_3$ 当且仅当 $(M)^{km} \models AG(p_1 \rightarrow AFp_2)$

PLTL 模型检测: 用 PLTL 模型检测算法，我们可证明

$$
\begin{aligned}
&(M)^{km} \models \Box(\neg(p_2 \wedge q_2)),\\
&(M)^{km} \models \Box(p_1 \wedge \neg(q_1 \vee q_2) \rightarrow (p_2 \, R \, \neg q_2)) \text{ 和}\\
&(M)^{km} \models \Box(p_1 \rightarrow \Diamond p_2) \text{ 。}
\end{aligned}
$$

因此 $M \models_I \varphi_1$ ， $M \models_I \varphi_2$ 和 $M \models_I \varphi_3$ 。

CTL 模型检测: 类似地，用 CTL 模型检测算法，我们亦可证明

$$
\begin{aligned}
&(M)^{km} \models AG(\neg(p_2 \wedge q_2)),\\
&(M)^{km} \models AG(p_1 \wedge \neg(q_1 \vee q_2) \rightarrow A(p_2 \, R \, \neg q_2)) \text{ 和}\\
&(M)^{km} \models AG(p_1 \rightarrow AFp_2) \text{ 。}
\end{aligned}
$$

因此 $M \models_I \varphi'_1$ ， $M \models_I \varphi'_2$ 和 $M \models_I \varphi'_3$ 。

模型检测算法过程举例: 由于计算的繁琐, 仅举一例, 即用不动点算法证明 $(M)^{km} \models AG(\neg(p_2 \wedge q_2))$, 即证明 $(M)^{km} \models \neg EF(p_2 \wedge q_2)$ 如下。

- 我们有 $EF(p_2 \wedge q_2) = \mu Z.((p_2 \wedge q_2) \vee EXZ)$ 。
- 为方便书写， $a(i), b(i), x(i), y(i), t(i)$ 分别写成 a, b, x, y, t 。

  定义 f_i 为以下表格中行数标号为数字 i 的状态集合。

标号	状态集合	说明
1	$\{w_i \mid a = 2 \wedge b = 2\}$	
11	$\{w_i \mid a = 1 \wedge (x = 0 \vee t = 0) \wedge b = 2\}$	
12	$\{w_i \mid a = 2 \wedge b = 1 \wedge (y = 0 \vee t = 1)\}$	
111	$\{w_i \mid (a = 0 \vee a = 3) \wedge x = 0 \wedge b = 2\}$	
112	$\{w_i \mid a = 1 \wedge (x = 0 \vee t = 0) \wedge b = 1 \wedge (y = 0 \vee t = 1)\}$	
121	$\{w_i \mid a = 1 \wedge (x = 0 \vee t = 0) \wedge b = 1 \wedge (y = 0 \vee t = 1)\ \}$	等同于 112
122	$\{w_i \mid a = 2 \wedge (b = 0 \vee b = 3) \wedge y = 0\}$	
1111	$\{w_i \mid a = 2 \wedge x = 0 \wedge b = 2\}$	包含于 1
1112	$\{w_i \mid (a = 0 \vee a = 3) \wedge x = 0 \wedge b = 1 \wedge (y = 0 \vee t = 1)\}$	
1121	$\{w_i \mid (a = 0 \vee a = 3) \wedge x = 0 \wedge b = 1 \wedge (y = 0 \vee t = 1)\}$	等同于 1112
1122	$\{w_i \mid a = 1 \wedge (x = 0 \vee t = 0) \wedge (b = 0 \vee b = 3) \wedge y = 0\}$	等同于 1221
1221	$\{w_i \mid a = 1 \wedge (x = 0 \vee t = 0) \wedge (b = 0 \vee b = 3) \wedge y = 0\}$	
1222	$\{w_i \mid a = 2 \wedge b = 2 \wedge y = 0\}$	包含于 1
11121	$\{w_i \mid a = 2 \wedge x = 0 \wedge b = 1 \wedge (y = 0 \vee t = 1)\}$	包含于 12
11122	$\{\}$	
12211	$\{\}$	
12212	$\{w_i \mid a = 1 \wedge (x = 0 \vee t = 0) \wedge b = 2 \wedge y = 0\}$	包含于 11

- 我们有 $(p_2 \wedge q_2) = \{w_i \mid a(i) = 2 \wedge b(i) = 2\} = f_1$ 。
- 设 $\tau(Z) = (p_2 \wedge q_2) \vee EXZ$ 。则 $\mu Z.\tau(Z)$ 的计算如下。

  $\tau^0(false) = false$;

  $\tau^1(false) = f_1$;

  $\tau^2(false) = \tau^1(false) \cup f_{11} \cup f_{12}$;

  $\tau^3(false) = \tau^2(false) \cup f_{111} \cup f_{112} \cup f_{122}$;

  $\tau^4(false) = \tau^3(false) \cup f_{1112} \cup f_{1221}$;

  $\tau^5(false) = \tau^4(false)$;

  因此 $EF(p_2 \wedge q_2) = \mu Z.((p_2 \wedge q_2) \vee EXZ) = \mu Z.\tau(Z) = \tau^4(false)$ 。

  因此 $AG(\neg(p_2 \wedge q_2)) = \neg EF(p_2 \wedge q_2) = S \setminus \tau^4(false)$ 。
- 由 $\tau^4(false)$ 不含有 $(M)^{km}$ 的初始状态知 $I \subseteq AG(\neg(p_2 \wedge q_2))$, 即 $(M)^{km} \models AG(\neg(p_2 \wedge q_2))$ 。

§9.1.3 基于模型检测工具的自动验证

鉴于手工验证的繁琐，我们可使用模型检测工具进行验证。

应用 VERDS 建模语言 VML 建立 VERDS 验证模型如下。

```
/*************************************************************************/
VVM         mutex1.vvm
VAR         a: {s0,s1,s2,s3}; b: {t0,t1,t2,t3}; x: 0..1; y: 0..1; t: 0..1;
INIT        a=s0; b=s0; x=0; y=0;
TRANS       a=s0:                          (y,t,a):=(1,1,s1);
            a=s1&(x=0|t=0):                (a):=(s2);
            a=s2:                          (y,a):=(0,s3);
            a=s3:                          (y,t,a):=(1,1,s1);
            b=t0:                          (x,t,b):=(1,0,t1);
            b=t1&(x=0|t=1):                (b):=(t2);
            b=t2:                          (x,b):=(0,t3);
            b=t3:                          (x,t,b):=(1,0,t1);
SPEC        AG(!(a=s2&b=s2));
            AG(!(a=s1&!(b=t1|b=t2))|A(a=s2 R b!=t2));
            AG(!(a=s1)|AF(a=s2));
/*************************************************************************/
```

应用 VERDS 验证 $\varphi_1', \varphi_2', \varphi_3'$ 的验证结果说明这些性质成立。

§9.1.4 公平约束模型的模型检测

假定我们的模型增加两条以下迁移，记为 T_1 。

$$\boxed{\begin{array}{l} a = s_2 \longrightarrow (a) := (s_2) \\ b = t_2 \longrightarrow (b) := (t_2) \end{array}}$$

这两条迁移表示当进程 A 在 s_2 状态时可以回到 s_2 这个状态，同样，进程 B 在 t_2 状态时可以回到 t_2 这个状态。记修改后的模型为 $M' = \langle T \cup T_1, \Theta \rangle$ 。

那么修改后的模型满足 φ_1 和 φ_2 ，不满足 φ_3 。

应用 VERDS 建模语言 VML 建立 VERDS 验证模型并验证 $\varphi_1', \varphi_2', \varphi_3'$ ，验证结果说明前两个性质成立和后一个性质不成立。

公平约束: 以 M' 为出发点，我们希望进程 A 在 s_2 状态重复有限次之后必须从 s_2 的状态出去且进程 B 在 t_2 状态重复有限次之后必须从 t_2 的状态出去。这个可以表示为公平约束集合 $\Phi = \{\neg(a = s_2), \neg(b = t_2)\}$ 。

那么这个公平卫式迁移模型为 $M'' = \langle T \cup T_1, \Theta, \Phi \rangle$ 。

应用 VERDS 建模语言 VML 的模块化建模功能建立 VERDS 验证模型如下。

```
/*************************************************************************/
VVM         mutex2.vvm
VAR         x: 0..1; y: 0..1; t: 0..1;
INIT        x=0; y=0;
PROC        p0: m0(x,y,0); p1: m0(y,x,1);
```

```
SPEC        AG(!(p0.a=s2&p1.a=s2));
            AG(!(p0.a=s1&!(p1.a=s1|p1.a=s2)) | A(p0.a=s2 R p1.a!=s2));
            AG(!(p0.a=s1) | AF(p0.a=s2));
MODULE      m0(x0,y0,k)
VAR         a: {s0,s1,s2,s3};
INIT        a=s0;
TRANS       a=s0:                   (y0,t,a):=(1,1-k,s1);
            a=s1&(x0=0|t=k):        (a):=(s2);
            a=s2:                   (y0,a):=(0,s3);
            a=s2:                   (a):=(s2);
            a=s3:                   (y0,t,a):=(1,1-k,s1);
FAIRNESS    !(a=s2);
/*************************************************************************/
```

应用 VERDS 验证 $\varphi_1', \varphi_2', \varphi_3'$ ，验证结果说明这些性质在新模型下成立，即 $M'' \models_I \varphi_1'$ ，$M'' \models_I \varphi_2'$ 和 $M'' \models_I \varphi_3'$ 。

§9.2 整数平方根算法

考虑用加法运算实现求一个数的整数平方根的算法。

整数平方根算法的卫式迁移模型: 给定 $B = (\{s_0, s_1, s_2, s_3, s_4, 0, 1, 2, +, *\}, \{=, \leq\})$ 。给定 $I = (Nat, I_0)$ 为 B 在自然数上的通常解释且 s_i 解释为 i 。

给定 $V = \{x, y_1, y_2, y_3, pc\}$ 。

(B, V) 上的整数平方根算法的卫式迁移模型定义为 $M = \langle T, \Theta \rangle$ 其中 Θ 为 $a = s_0$ 且 T 为以下迁移的集合：

$pc = s_0$	$\longrightarrow$	$(y_1, y_2, y_3, pc) := (0, 1, 1, s_1)$
$pc = s_1 \wedge (y_3 \leq x)$	$\longrightarrow$	$(pc) := (s_2)$
$pc = s_1 \wedge \neg(y_3 \leq x)$	$\longrightarrow$	$(pc) := (s_4)$
$pc = s_2$	$\longrightarrow$	$(y_1, y_2, pc) := (y_1 + 1, y_2 + 2, s_3)$
$pc = s_3$	$\longrightarrow$	$(y_3, pc) := (y_3 + y_2, s_1)$

整数平方根算法的性质: 我们考虑以下两个性质。

$\varphi_1 = (x > 0 \rightarrow \Box(pc = s_4 \rightarrow y_1 = \sqrt{x}))$ 。
$\varphi_2 = (x > 0 \rightarrow \Diamond(pc = s_4))$ 。

其中 $a = \sqrt{b}$ 表示 $a^2 \leq b \wedge b < (a+1)^2$ 。这两个 PLTL 性质亦可用 CTL 表示如下。

$\varphi_1' = (x > 0 \rightarrow AG(pc = s_4 \rightarrow y_1 = \sqrt{x}))$ 。
$\varphi_2' = (x > 0 \rightarrow AF(pc = s_4))$ 。

§9.2.1 推理验证

证明: $M \models_I \varphi_1$ 。

应用 §7.3 节的证明结论形式为 $\varphi \Rightarrow \Box\psi$ 的证明规则， $M \models_I \varphi_1$ 可根据以下步骤证明。

首先证明 $M \models_I \Box(\Theta \wedge x > 0 \rightarrow \Box(pc = s_4 \rightarrow y_1 = \sqrt{x}))$ 如下。

定义 $\varphi = (\Theta \wedge x > 0)$ 且 $\psi = (pc = s_4 \rightarrow y_1 = \sqrt{x})$ 。
定义 ϕ 为以下公式的析取。

$$
\begin{aligned}
&pc = s_0 \wedge (x > 0)\\
&pc = s_1 \wedge (y_1^2 \leq x \wedge y_2 = 2 * y_1 + 1 \wedge y_3 = (y_1 + 1)^2)\\
&pc = s_2 \wedge ((y_1 + 1)^2 \leq x \wedge y_2 = 2 * y_1 + 1 \wedge y_3 = (y_1 + 1)^2)\\
&pc = s_3 \wedge (y_1^2 \leq x \wedge y_2 = 2 * y_1 + 1 \wedge y_3 = y_1^2)\\
&pc = s_4 \wedge (y_1 = \sqrt{x})
\end{aligned}
$$

则有 $\varphi \Rightarrow \phi$ 且 $\phi \rightarrow [T]\phi$ 且 $\phi \Rightarrow \psi$ 。
因而根据证明规则我们有 $\varphi \Rightarrow \Box\psi$ ，即 $M \models_I \Box(\Theta \wedge x > 0 \rightarrow \Box\psi)$ 。
因而 $M \models_I \Theta \wedge x > 0 \rightarrow \Box\psi$ 。进而我们有 $M \models_I \varphi_1$ 。

证明: $M \models_I \varphi_2$ 。
应用 §7.3 节的结论形式为 $\varphi \Rightarrow \Diamond\psi$ 的证明规则， $M \models_I \varphi_3$ 可根据以下步骤证明。
首先证明 $M \models_I \Box((pc = s_0 \wedge x > 0) \rightarrow \Diamond(pc = s_4))$ 如下。
扩展 B 使得 $B' = (F \cup \{f\}, P)$ 。扩展 I_0 使得以下成立。

$$
\begin{aligned}
&I(f(s_0, x, y_3)) = 3x + 1\\
&I(f(s_i, x, y_3)) = 3(x + 1 - y_3) + 1 - i \qquad (i = 1, 2, 3)\\
&I(f(s_4, x, y_3)) = 0
\end{aligned}
$$

定义 $\varphi, \psi, W, w, e, \phi$ 如下。

$$
\begin{aligned}
\varphi &= (pc = s_0 \wedge x > 0)\\
\psi &= (pc = s_4)\\
W &= (Nat, \leq)\\
w &= true\\
e &= f(pc, x, y_3)\\
\phi &= (pc \neq s_0 \rightarrow y_2 \geq 1) \wedge (pc = s_2 \vee pc = s_3 \rightarrow y_3 \leq x)
\end{aligned}
$$

则 $\varphi \Rightarrow (\psi \vee \phi)$ 且 $\phi \Rightarrow w_x^e \wedge (\psi \vee E(T))$ 且 $\phi \wedge e = v \rightarrow [T](\psi \vee (\phi \wedge e < v))$ 。
因而根据证明规则我们有 $\varphi \Rightarrow \Diamond\psi$ ，即 $M \models_I \Box((pc = s_0 \wedge x > 0) \rightarrow \Diamond(pc = s_4))$ 。
因而 $M \models_I (pc = s_0 \wedge x > 0) \rightarrow \Diamond(pc = s_4)$ 。进而我们有 $M \models_I \varphi_2$ 。

整数论域: 若 B 改在整数论域上解释，则在验证 $M \models_I \varphi_2$ 时，各元素的定义和选取会略有不同。首先可修改 f 的定义如下。

$$
\begin{aligned}
&I(f(s_0, x, y_2, y_3)) = 3x + 3\\
&I(f(s_i, x, y_2, y_3)) = 3(x + 1 + y_2 - y_3) - i \quad (i = 1, 2)\\
&I(f(s_3, x, y_2, y_3)) = 3(x + 1 + y_2 - y_3) - 9\\
&I(f(s_4, x, y_2, y_3)) = 0
\end{aligned}
$$

然后可选取 $\varphi, \psi, W, w, e, \phi$ 如下。

$$
\begin{array}{|lcl|lcl|}
\hline
\varphi & = & (pc = s_0 \wedge x > 0) & \phi & = & (x \geq 0) \wedge \\
\psi & = & (pc = s_4) & & & (pc = s_1 \rightarrow y_2 \geq 1 \wedge y_3 \leq x + y_2) \wedge \\
W & = & (Nat, \leq) & & & (pc = s_2 \rightarrow y_2 \geq 1 \wedge y_3 \leq x) \wedge \\
w & = & (x \geq 0) & & & (pc = s_3 \rightarrow y_2 \geq 3 \wedge y_3 \leq x) \\
e & = & f(pc, x, y_2, y_3) & & & \\
\hline
\end{array}
$$

其余部分的证明类似自然数解释下的证明。

§9.2.2 基于模型检测工具的自动验证

由于模型检测算法主要是基于穷尽搜索，我们需要对变量的取值范围进行限制。

自动验证: 设变量的值域范围为 $\{0, 1, ..., 30\}$ 。应用 VERDS 建模语言 VML 建立 VERDS 验证模型如下。

```
/************************************************************************/
VVM         isqrt1.vvm
VAR         pc:{s0,s1,s2,s3,s4}; x:0..30; y1:0..30; y2:0..30; y3:0..30;
INIT        pc=s0; x<=30; y1=0; y2=0; y3=0;
TRANS       pc=s0:                  (y1,y2,y3,pc):=(0,1,1,s1);
            pc=s1&(y3<=x):          (pc):=(s2);
            pc=s1&!(y3<=x):         (pc):=(s4);
            pc=s2:                  (y1,y2,pc):=(y1+1,y2+2,s3);
            pc=s3:                  (y3,pc):=(y3+y2,s1);
SPEC        !(x>0) | AG(!(pc=s4) | (x>=y1*y1) & x<(y1+1)*(y1+1));
            !(x>0) | AF(pc=s4);
/************************************************************************/
```

应用 VERDS 验证 φ_1', φ_2'，验证结果说明这些性质 (在变量取值范围受限的情况下) 成立。

§9.2.3 错误检查与模型检测中的反例生成

假定我们在模型中把条件 $y_3 \leq x$ 误写为 $y_3 < x$ 。 应用 VERDS 建模语言 VML 建立修改的 VERDS 验证模型 (即将两处 $y_3 \leq x$ 改为 $y_3 < x$) 并验证 φ_1', φ_2'，则有以下结果： φ_2' 成立而 φ_1' 不成立。验证结果说明这个错误影响计算结果的正确性，但不影响模型运行的终止性。

反例生成: 对前面不成立的性质 VERDS 可提供反例，即不满足性的见证。整理后可得如下不满足 φ_1' 的运行路径。

状态	pc	x	y1	y2	y3
0	s0	1	0	0	0
1	s1	1	0	1	1
2	s4	1	0	1	1

这条运行路径显示开始时 x 的值为 1 ，最后 $pc = s4$ 时 $y1$ 的值为 0 ，而此时的 $y1$ 值应为 1 。因而模型不满足 $x > 0 \rightarrow AG((x >= y1 * y1)\&x < (y1 + 1) * (y1 + 1))$ 。若改为验证 $x > 5 \rightarrow AG((x >= y1 * y1)\&x < (y1 + 1) * (y1 + 1))$ ，那么使得性质不满足的最小的 x 的值为 9 。模型检测工具 VERDS 所提供的反例路径如下。

状态	pc	x	y1	y2	y3
0	s0	9	0	0	0
1	s1	9	0	1	1
2	s2	9	0	1	1
3	s3	9	1	3	1
4	s1	9	1	3	4
5	s2	9	1	3	4
6	s3	9	2	5	4
7	s1	9	2	5	9
8	s4	9	2	5	9

这条运行路径显示开始时 x 的值为 9，最后 $pc = s4$ 时 $y1$ 的值为 2，而此时的 $y1$ 值应为 3。因而模型不满足 $x > 5 \rightarrow AG((x >= y1 * y1)\&x < (y1 + 1) * (y1 + 1))$。

§9.3 查找模型中的特定状态和路径

特定状态: 特定状态的查找问题就是问是否存在可达的满足给定性质的状态。

设给定性质表示为状态公式 φ，则该问题可转化为检查模型是否满足 $AG(\neg\varphi)$。若所检查的性质不满足，则通过查看反例可获得满足 φ 的一个可达状态，以及模型运行是怎么到达这个状态的。

模型检测可在很大状态空间中查找具有特定性质的可达状态。应用模型检测工具 VERDS，对于一些简单的模型和性质，搜索的可达状态空间可以达到 10^{50} 个状态。

特定路径: 特定路径的查找问题就是问是否存在计算路径满足给定的性质。

设给定性质表示为 $\Diamond(\varphi)$ 其中 φ 为状态公式,则该问题亦可转化为检查模型是否满足 $AG(\neg\varphi)$。若所检查的性质不满足，则通过查看反例可获得一条满足 $\Diamond(\varphi)$ 的路径。

通常情况下，路径由于可以任意长，路径空间是无穷的，那么其中可能有一小部分路径满足给定性质。模型检测工具可以应用于搜索小概率的满足一定性质的路径。

§9.4 练习

1. 应用加减法运算设计模 2 算法。应用推理方法证明该算法正确性。将变量取值范围进行适当限制，用模型检测工具验证算法正确性。

2. 设计 3 个进程的环选举算法并构造迁移模型。证明（或应用工具辅助证明）该算法满足安全性质（每一个进程要么不知道结果要么知道正确的结果）和必达性质（选举算法总会有结果且每一个进程都知道结果）。

附录 A　建模语言 VML

本附录介绍基于卫式迁移模型的建模语言 VML 。
用 VML 写的模型称为 VVM 模型，即 VERDS 验证模型。

```
1. Atoms

An atom may be any sequence of characters starting with a character in
the set {a-z_} and followed by a possibly empty sequence of characters
belonging to the set {a-z0-9_}. A number is any sequence of digits. A digit
belongs to the set {0-9}.

All characters and case in a name are significant. Whitespace characters
are space (SPACE), tab (TAB) and newline (RET). Comments in verds language
is any string starting with two slashes (‘//’) and ending with a newline.
Any other tokens recognized by the parser are enclosed in quotes in the
syntax expressions below. Grammar productions enclosed in square brackets
(‘[]’) are optional.

2.  Expressions

Expressions are constructed from variables, constants, and a collection
of operators, including temporal operators, Boolean connectives, and
integer arithmetic operators.

2.1.    Numeric Expressions

Numeric expressions are expressions built only from current state
variables. The syntax of numeric expressions is as follows:

numeric_expr ::
          symb_const                        ;; a symbolic constant
        | numb_const                        ;; a numeric constant
        | variable_id                       ;; a variable identifier
        | "(" numeric_expr ")"
        | numeric_expr "+" numeric_expr     ;; integer addition
        | numeric_expr "-" numeric_expr     ;; integer subtraction
        | numeric_expr "*" numeric_expr     ;; integer multiplication
        | numeric_expr "/" numeric_expr     ;; integer division
        | numeric_expr "%" numeric_expr     ;; integer remainder

symb_const ::    atom

numb_const ::    number
```

```
variable_id ::    atom | atom"."atom | variable_id "[" numeric _expr "]"
```

A symbolic constant is represented by an atom, which are predefined ones, including the atom "pid". Such constants are meant to be used in particular sections of the declarations.

A numeric constant is represented by a number. A simple variable identifier is either an atom representing a simple variable or such a variable identifier preceded by a process identifier (represented by the first atom in the construct atom"."atom) in order to refer a local variable of a process. A simple variable identifier followed by a sequence of indices represents an element of an array variable.

The order of parsing precedence for operators from high to low is:

*,/,%
+,-

Operators of equal precedence associate to the left. Parentheses may be used to group expressions.

2.2. Logic Expressions

Logic expressions are expressions built from numeric expressions. Logic expressions can be used to specify sets of states, e.g. the initial set of states. The syntax of logic expressions is as follows:

```
logic_expr ::
          "TRUE"                          ;; The boolean constant 1
        | "FALSE"                         ;; The boolean constant 0
        | logic_const                     ;; predefined constants
        | "(" logic _expr ")"
        | numeric_expr "=" numeric_expr   ;; equality
        | numeric_expr "!=" numeric_expr  ;; inequality
        | numeric_expr "<" numeric_expr   ;; less than
        | numeric_expr ">" numeric_expr   ;; greater than
        | numeric_expr "<=" numeric_expr  ;; less than or equal
        | numeric_expr ">=" numeric_expr  ;; greater than or equal
        | logic_expr "&" logic_expr       ;; logical and
        | logic_expr "|" logic_expr       ;; logical or
        | "!" logic_expr                  ;; logical not

logic_const ::    atom
```

A logical constant is represented by an atom, which are predefined ones, including the atom "running". Such constants are meant to be used in particular sections of the declarations.

The order of parsing precedence for operators from high to low is:

```
=,!=,<,>,<=,>=
!
&
|
```

Operators of equal precedence associate to the left. Parentheses may be used to group expressions.

2.3. Temporal Logic Expressions

Temporal logic expressions are expressions built from logic expressions. Logic expressions can be used to specify properties of the finite state machine. The syntax of temporal logic expressions is as follows:

```
temporal_expr ::
          logic_expr
        | "AX" temporal_expr
        | "AG" temporal_expr
        | "AF" temporal_expr
        | "A" "(" temporal_expr "R" temporal_expr ")"
        | "A" "(" temporal_expr "U" temporal_expr ")"
        | "EX" temporal_expr
        | "EG" temporal_expr
        | "EF" temporal_expr
        | "E" "(" temporal_expr "R" temporal_expr ")"
        | "E" "(" temporal_expr "U" temporal_expr ")"
        | temporal_expr "&" temporal_expr              ;; logical and
        | temporal_expr "|" temporal_expr              ;; logical or
        | "!" temporal_expr                            ;; logical not
```

3. Discrete Transition Systems

A discrete transition system consists three parts: the variables representing the state of the system, the initial states, and the transition relation. In addition, property specifications may be attached to a discrete transition system, in order to check whether the discrete transition system is correct with respect to the properties. The

transition relation may be divided into different modules.

3.1. State Variables

A state of the model is an assignment of values to a set of state variables. These variables are declared by the notation:

```
var_declaration :: variable ":" type ";" ... variable ":" type ";"

variable ::  atom | variable "[" number ".." number "]"

type ::  number ".." number | "{" atom "," atom "," ... "," atom "}"
```

A variable is either a simple variable or an array variable. The type associated with a variable declaration can be either a scalar or an enumeration of a set of atoms.

Two enumeration types must either be identical or with disjoint sets of elements.

A variable of type Boolean may be represented by a variable of type 0..1 such that x=1 means x for the Boolean variable x, x=0 means not x, and negating the Boolean variable x may be implemented by the arithmetic expression 1-x.

3.2. Initial Values of State Variables

Initial values of state variables are specified by a list of formulas constraining the state variables. As a special case, the initial values of state variables may be specified as a list of formulas of the form x=c where x is a variable identifier and c is a constant. The initial values are declared by the notation:

```
init_declaration :: logic_expr ";" ... logic_expr ";"
```

3.3. Transition Relations

The transition relation of a process is implemented by a set of transition relations. The transition relation is declared by the notation:

```
trans_declaration :: cond ":" assignment ";" ... cond ":" assignment ";"

cond  :: logic_expr
```

```
assignment ::
    "(" variable_id "," variable_id "," ... variable_id ")"
    ":="
    "(" numeric_expr "," numeric_expr "," ... numeric_expr ")" ";"
```

3.4. Fairness Declarations

The transition relation may be augmented with fairness requirements that impose restrictions on the valid execution paths. The specification is declared by the notation:

```
fairness_declaration  :: logic_expr ";" ... logic_expr ";"
```

The logical constant "running" is a special proposition that may be used in a fairness declaration. The proposition is satisfied when the process in which the fairness is declared is executed.

3.5. Specification Declarations

The specification of system and process properties is represented by a temporal logic expression. The specification is declared by the notation:

```
spec_declaration :: temporal_expr ";" ... temporal_expr ";"
```

3.6. Module Declarations

A module is an encapsulated collection of declarations. Once defined, a module can be reused as many times as necessary. Modules can also be so that each instance of a module can refer to different data values.

The syntax of a module declaration is as follows.

```
module ::
        "MODULE"    atom "(" [ par_list ] ")"
        "VAR"       [ var_declaration ]
        "INIT"      init_declaration
        "TRANS"     trans_declaration
     [ "FAIRNESS"   fairness_declaration ]

par_list :: atom | atom "[]" | par_list "," par_list
```

The atom immediately following the keyword "MODULE" is the name associated with the module. The optional par_list in parentheses is the list of the formal parameters of the module. The symbols "[]" indicates that the atom

preceding the symbols represents an array variable. Whenever these parameters occur in expressions within the module, they are replaced by the actual parameters which are supplied when the module is instantiated.

The symbolic constant "pid" is a special constant that may be used in a module declaration. The constant is interpreted as the numerical constant representing the id of the process (the first process has id=0), which is an instance of the module.

3.7. Transition system declarations

A discrete transition system may have one or more processes that are instances of one or more module declarations. The syntax of a discrete transition system (called verds verification model) declaration is as follows.

```
verds_verification_model ::
        "VVM"       [ atom ]
    [ "DEFINE"      definition_declaration ]
        "VAR"       [ var_declaration ]
        "INIT"      init_declaration
        "PROC"      process_declaration
        [ "FAIRNESS"    fairness_declaration ]
        [ "SPEC"    spec_declaration ]

definition_declaration ::
    string_latin_letters "=" string_visible_characters "\n"
    ...
    string_latin_letters "=" string_visible_characters "\n"

process_declaration ::
    atom ":" atom "(" [ par_list ] ")" ";"
    ...
    atom ":" atom "(" [ par_list ] ")" ";"
```

The optional atom after the keyword VVM is the name of the discrete transition system.

A list of definitions may be provided. A defined word (which is a string of Latin letters) is an abbreviation of the defining string of visible characters, and the end-line symbol indicates the end of the definition. If a defining string is long, a backslash '\' may be used to ignore the end-line character immediately after this backslash.

A process is an instance of a module. The first atom in the process declaration is the name of the process (or the module instance), the second atom is the name of a module and the atoms in the optional list are parameters passed to the module.

The variables in the var_declaration section are global variables that can be used in all modules.

The complete verification model consists of a verds_verification_model declaration followed by a list of module declarations that are referred to in the process declaration.

For simplicity, a single process discrete transition system may be declared by the following syntax as well.

```
single_process_verds_verification_model ::
        "VVM"       [ atom ]
        "VAR"       [ var_declaration ]
        "INIT"      init_declaration
        "TRANS"     trans_declaration
        [ "FAIRNESS"    fairness_declaration ]
        [ "SPEC"    spec_declaration ]
```

This declaration is similar to the previous declaration of a VVM with the difference that the process declaration is replaced by a transition declaration.

参考文献

1. Alur, R., Dill, D.L.: A theory of timed automata. Theor. Comput. Sci. 126(2), 183-235 (1994)

2. Ben-Ari, M., Manna, Z., Pnueli, A.: The Temporal Logic of Branching Time. In: POPL'81. pp. 164-176 (1981)

3. Bhat, G., Cleaveland, R.: Efficent Local Model-Checking for Fragments of the Modal mu-Calculus. In: TACAS'96. pp. 107-126 (1996)

4. Biere, A., Cimmatti, A., Clarke, E., Strichman, O., , Zhu, Y.: Bounded model checking. Advances in Computers 58, 117-148 (2003)

5. Biere, A., Cimmatti, A., Clarke, E.M., Zhu, Y.: Symbolic Model Checking without BDDs. In: TACAS'99. pp. 193-207 (1999)

6. Bradfield, J., Esparza, J., Mader, A.: An Effective Tableau System for the Linear Time *mu*-Calculus. In: ICALP'97. pp. 98-109 (1997)

7. Buccafurri, F., Eiter, T., Gottlob, G., Leone, N.: On actl formulas having linear counterexamples. J. Comput. Syst. Sci. 62, 463-515 (2001)

8. Burch, J.R., Clarke, E.M., McMillan, K.L., Dill, D.L., Hwang, J.: Symbolic model checking: 10^{20} states and beyond. In: LICS'90. pp. 428-439 (1990)

9. Clark, E.M., Grumberg, O., Peled, D.: Model Checking. MIT press (1999)

10. Clarke, E.M., Emerson, E.A.: Design and Synthesis of Synchronization Skeletons Using Branching-Time Temporal Logic. In: Logic of Programs 1981. pp. 52-71 (1981)

11. Courcoubetis, C., Vardi, M., Wolper, P., Yannakakis, M.: Memory-efficient algorithms for the verification of temporal properties. Formal Methods in System Design 1(2-3), 275-288 (1992)

12. Dax, C., Hofmann, M., Lange, M.: A Proof System for the Linear Time *mu*-Calculus. In: FSTTCS'06. pp. 98-109 (2006)

13. Demri, S., Schnoebelen, P.: Design and Synthesis of Synchronization Skeletons Using Branching-Time Temporal Logic. In: STACS'98. pp. 61-72 (1998)

14. Emerson, E.A.: Formal models and semantics. Temporal and Modal Logic pp. 995-1072 (1990)

15. Emerson, E.A., Halpern, J.Y.: "sometimes" and "not never" revisited: on branching versus linear time temporal logic. J. ACM 33(1), 151-178 (1986)

16. Francez, N.: Program verification. Addison-Wesley Publishing Company Inc. (1992)

17. Gabbay, D.M., Pnueli, A., Shelah, S., Stavi, J.: On the temporal analysis of fairness. In: POPL'80. pp. 163-173 (1980)

18. Henzinger, T.A.: The Theory of Hybrid Automata. In: LICS'96. pp. 278-292 (1996)

19. Hoare, C.A.R.: An axiomatic basis for computer programming. Commun. ACM 12, 576-580 (1969)

20. Holzmann, G.J.: Design and Validation of Computer Protocols. Prentice Hall (1990)

21. Holzmann, G.J.: The SPIN Model Checker. Addison-Wesley (2003)

22. Kaivola, R.: A Proof System for the Linear Time *mu*-Calculus. In: CONCUR'95. pp. 423-437 (1995)

23. Kozen, D.: Results on the propositional *mu*-calculus. Theor. Comput. Sci. 27, 333-354 (1983)

24. Loeckx, J., Sieber, K.: The foundation of program verification. John Wiley & Sons Ltd. (1984)

25. Maidl, M.: The common fragment of CTL and LTL. In: FOCS'00. pp. 643-652 (2000)

26. Manna, Z., Pnueli, A.: How to cool a temporal proof system for your pet language. In: POPL'83. pp. 141-154 (1983)

27. Manna, Z., Pnueli, A.: Completing the temporal picture. Theor. Comput. Sci. 83, 97-130 (1991)

28. McMillan, K.L.: Symbolic Model Checking. Kluwer Publisher (1993)

29. Nielsen, M., Winskel, G.: Petri nets and bisimulation. Theor. Comput. Sci. 153(1- 2), 211-244 (1996)

30. Peled, D.A.: Software Reliability Methods. Springer (2001)

31. Pnueli, A.: The Temporal Logic of Programs. In: FOCS'77. pp. 46-57 (1977)

32. Ramsey, F.P.: On a problem of formal logic. Proceedings of the London Mathematical Society 30(1), 264-286 (1930)

33. Reynolds, M.: An axiomatization of full computation tree logic. The Journal of Symbolic Logic 66(3), 1011-1057 (2001)

34. Schneider, K.: Verification of reactive systems: formal methods and algorithms. Springer (2004)

35. Schnoebelen, P.: The complexity of temporal logic model checking. Advances in Modal Logic 4, 1-44 (2002)

36. Sistla, A.P., Clarke, E.M.: The complexity of propositional linear temporal logics. J. ACM 32(3), 733-749 (1985)

37. Tarjan, R.: Depth first search and linear graph algorithms. SIAM Journal on Computing 1(2), 146-160 (1972)

38. Vardi, M.Y.: A Temporal Fixpoint Calculus. In: POPL'88. pp. 250-259. ACM (1988)

39. Vardi, M.Y., Wolper, P.: An automata-theoretic approach to automatic program verification. In: LICS'86. pp. 332-344 (1986)

40. Zhang, W.: Bounded Semantics of CTL and SAT-based Verification. In: ICFEM'09. pp. 286-305 (2009)

41. Zhang, W.: QBF Encoding of Temporal Properties and QBF-Based Verification. In: IJCAR'14. pp. 286-305 (2014)

42. Zhang, W.: Bounded semantics. Theor. Comput. Sci. 564, 1-29 (2015)