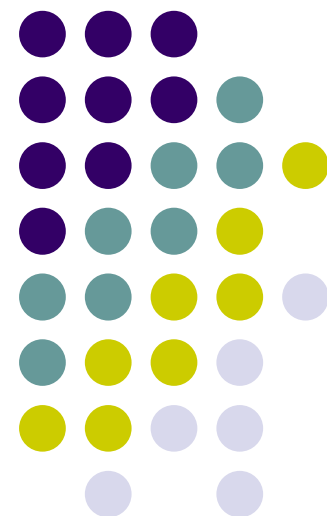




南京大學
Nanjing University

第12讲-模态逻辑概述





- 模态逻辑（Modal Logic）是一类最初由哲学家发展起来的用于研究真理的不同模式（mode）的逻辑。这些模式主要包括：
 - 可能与必然– 基本模态逻辑；
 - 过去与将来– 时态逻辑（Temporal Logic）；
 - 知道与相信– 认知逻辑（Epistemic Logic）；
 - 义务与允许– 道义逻辑（Deontic Logic）。
- 模态逻辑与计算机科学、人工智能均有密切的联系，
 - 时态逻辑– 模型检测（Model Checking）；
 - 认知逻辑– 知识表示（Knowledge Representation）；
 - 道义逻辑– 规范系统（Normative Systems）。



Blackburn, P. et al.(2002)将模态逻辑的特征总结为如下三点:

- 1) 模态逻辑是用于描述关系结构的简单而富于表达力的语言;
- 2) 模态逻辑为关系结构提供了一种内部和局部的视角;
- 3) 模态逻辑并不是孤立的形式化系统。

本讲将主要围绕上述三个特点，概述模态逻辑的基本语法语义。



关系结构

定义12.1. 关系结构是一个元组 $\mathfrak{F} = (W, R_1, \dots, R_n)$, 其中 W 被称为 \mathfrak{F} 的域 (Domain) 或宇宙 (Universe), R_1, \dots, R_n 是 W 上的关系。

- W 中的元素在许多不同的场景下通常具有不同的名称, 如: 点、状态、节点、世界、时间、瞬间、状况等等。
- 关系结构的一个有趣的特征是——它们通常可以被表示成简单的图形。



例12.1 严格偏序是一种关系结构。它是一个二元组 (W, R) ，其中 R 满足

- 反自反 $(\forall x \neg Rxx)$,
- 传递 $(\forall xyz (Rxy \wedge Ryz \rightarrow Rxz))$,
- 反对称 $(\forall xy \neg (Rxy \wedge Ryx))$.

一个严格偏序是一个线序（或全序）如果它也满足三分法条件 (trichotomy) : $\forall xy (Rxy \vee x = y \vee Ryx)$.

如图12.1 所示是严格偏序的一个例子，其中

- $W = \{1, 2, 3, 4, 6, 8, 12, 24\}$,
- Rxy 表示“ x 和 y 是不同的, 而且 y 可被 x 整除”.

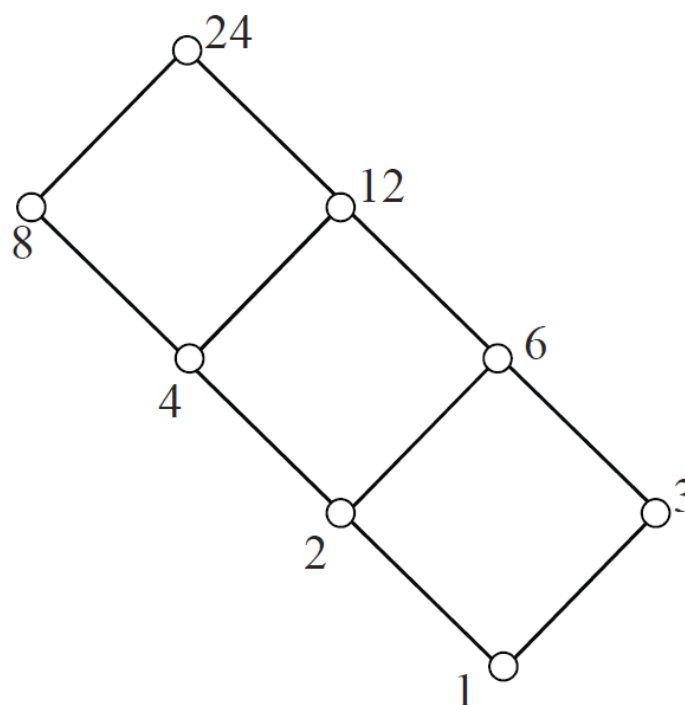


图 12.1: 一个严格偏序



例12.2. 标注转换系统 (Labeled Transition System, 简称LTS) 是一种在计算机科学中广为使用的简单关系结构, 定义为元组 $(W, \{R_a | a \in A\})$, 其中 W 是一个非空状态集, A 是一个非空的标注集, 而且对于任何 $a \in A$, $R_a \subseteq W \times W$ 。

转换系统可以被看作是一种计算的抽象模型:

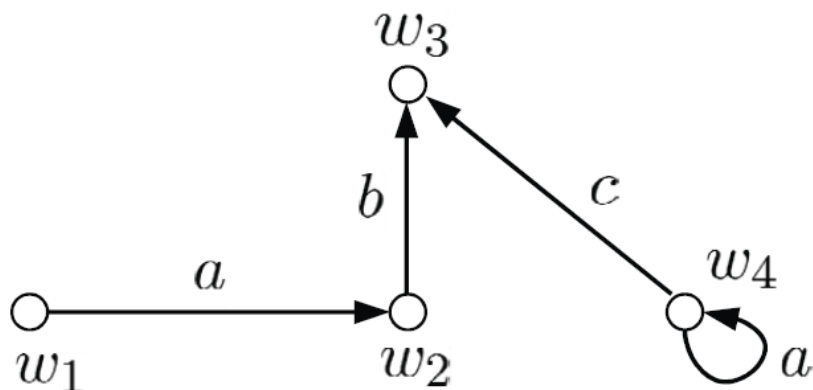


图 12.2: 一个确定转换系统

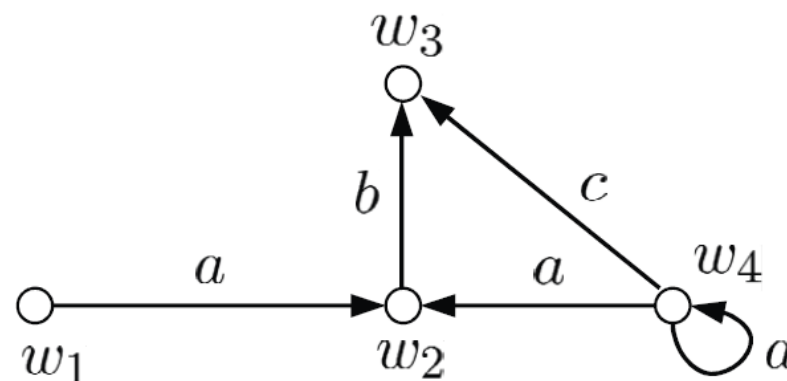


图 12.3: 一个非确定转换系统



例12.3. 时间的内在结构及其表示是一个耐人寻味的话题。通常可以假设时间是线性的，即

- i) 时间是离散的；
- ii) 有一个没有前驱的初始时刻； 并且
- iii) 有无穷的后续时刻进入未来。



在线性时间的假设下，时间的内在结构是一个全序集 $(S, <)$ ，并且可以进一步假定其内在结构同构于自然数集 $(\mathbb{N}, <)$ 。

这意味着可以把线性时间的结构定义为元组 (S, x) ，其中

- S 是一个状态集合；
- $x : \mathbb{N} \rightarrow S$ 是一个无穷的状态序列。

x 也叫做时间线 (timeline)，通常可被更简洁地表示为

$$x = (s_0, s_1, s_2, \dots) = (x(0), x(1), x(2), \dots)$$

此外，在一些场景中 x 也被叫做路径(path)、全路径(fullpath)、计算序列(computation sequence)或计算(computation)。

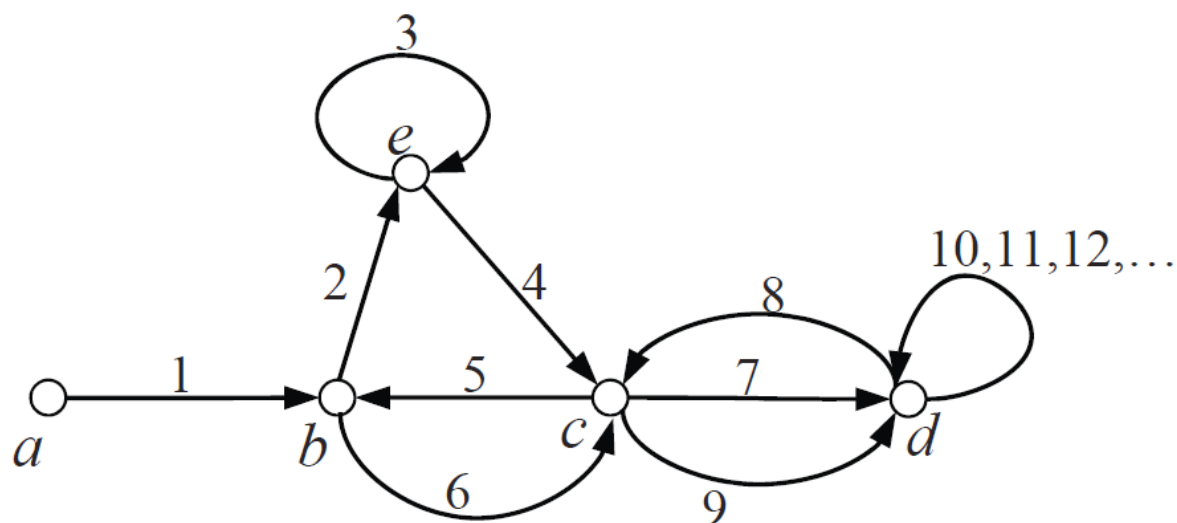


图 12.4：一个线性时间结构

如图12.4是一个线性时间结构：

$$S = \{a, b, c, d, e\}, x = (a, b, e, e, c, b, c, d, c, d, \dots)。$$

图中描述的系统以一个确定的顺序进行状态迁移。



而很多现实系统的运行具有不确定性，其中任何状态都仅有一个前驱状态，但是可能有多个不同的后续状态，这实质上对应于树状的时间结构。

这类时间结构可以表示为 (S, R) ，其中

- S 是一个状态集合；
- R 是一个定义在 S 上的完全的二元关系（即满足 $\forall s \in S, \exists t \in S : (s, t) \in R$ ）。

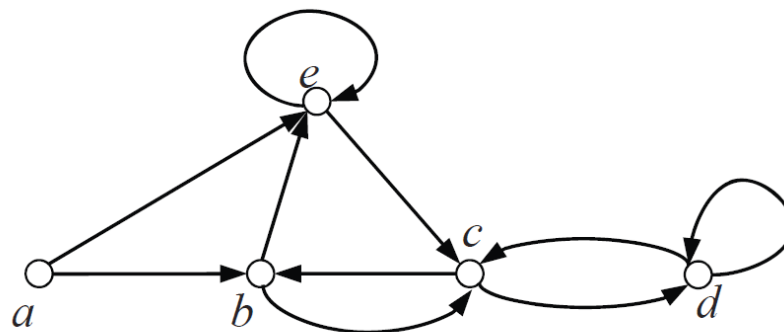


图 12.5 : 一个树状时间结构

如图12.5可表示一个树状时间结构，直观上看，其“树状”体现于从任意节点解开（unwind）均能得到一个树状结构。如图12.6.

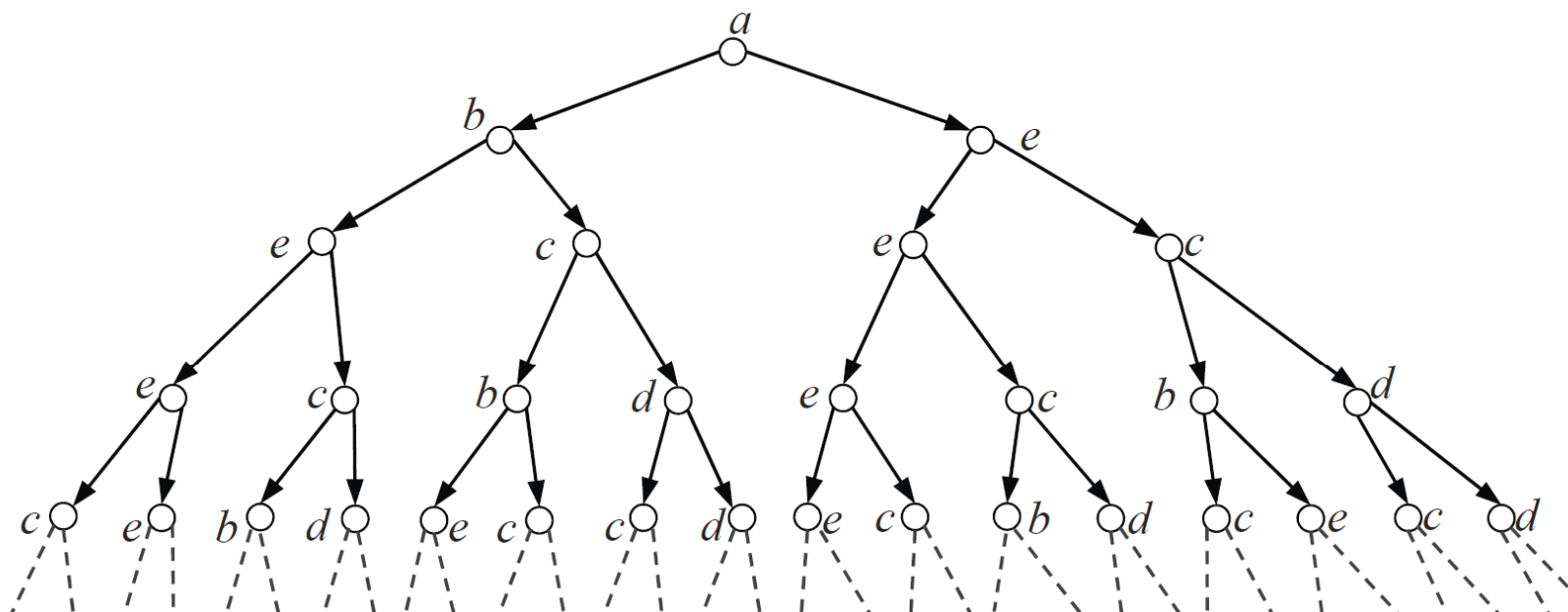


图 12.6 : 图12.5 表示的结构从状态a解开



基本模态逻辑

定义12.5. 基本模态逻辑的模型(model)为 $\mathfrak{M} = (W, R, L)$, 其中

- W 是一个非空集;
- R 是 W 上的一个关系;
- $L: W \rightarrow 2^\Phi$ 为标记函数, 把 W 中的各个点标记上在该点为真的命题符。

其中 Φ 是一个潜在的命题符的集合。

可以发现基本模态逻辑的模型是由一个关系结构 $\mathfrak{F} = (W, R)$ (通常称为框架 (frame)) 和一个标记函数 L 构成。



定义12.2. 基本的模态语言基于一个命题符的集合 Φ 以及一个一元模态算子 \diamond (“diamond”) 而定义, 它的合式公式 (well-formed formula) φ 由以下规则给出:

$$\varphi ::= p \mid \perp \mid \neg\varphi \mid \varphi \vee \psi \mid \diamond\varphi$$

其中 $p \in \Phi$, ψ 是一个合式公式。通常假定命题符的集合 Φ 是一个可数无穷集 $\{p_0, p_1, \dots\}$, 当然在某些情况下也能假定其为一个有穷集或不可数无穷集。

算子 \diamond 也有一个对偶算子 \square (“box”), 定义为 $\square\varphi := \neg\diamond\neg\varphi$.

$\diamond\varphi$ 通常读作“可能 φ ”, 那么 $\square\varphi$ 应该读作“不可能不 φ ”, 即“必然 φ ”。



例12.4. 下面给出一些基本模态逻辑的合式公式:

$$\mathbf{K}: \Box(\varphi \rightarrow \psi) \rightarrow (\Box\varphi \rightarrow \Box\psi)$$

$$\mathbf{T}: \Box\varphi \rightarrow \varphi$$

$$\mathbf{4}: \Box\varphi \rightarrow \Box\Box\varphi \quad (\text{or} \quad \Diamond\Diamond\varphi \rightarrow \Diamond\varphi)$$

$$\mathbf{B}: \varphi \rightarrow \Box\Diamond\varphi$$

$$\mathbf{D}: \Box\varphi \rightarrow \Diamond\varphi$$

$$\mathbf{5}: \Diamond\varphi \rightarrow \Box\Diamond\varphi$$

其中 φ, ψ 是命题符或一般的合式公式。



基于关系结构定义的语义是1950年代由Saul Kripke提出的，现在被广泛被用于时态逻辑及模型检测。

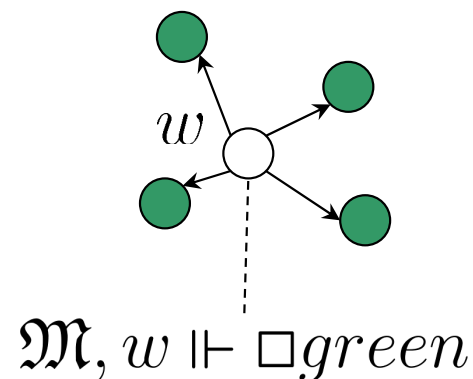
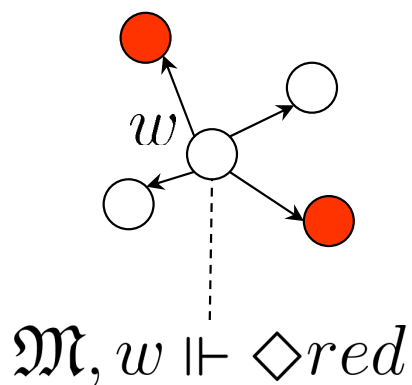
定义12.6. 令 w 是模型 $\mathfrak{M} = (W, R, L)$ 中的任意状态。一个基本模态语言公式 φ 在状态 w 被满足（或为真），表示为 $\mathfrak{M}, w \Vdash \varphi$ ，可归纳地被定义如下：

- $\mathfrak{M}, w \Vdash p$ 当且仅当 $p \in L(w)$, 其中 $p \in \Phi$;
- $\mathfrak{M}, w \Vdash \perp$ 从不成立;
- $\mathfrak{M}, w \Vdash \neg\varphi$ 当且仅当 $\mathfrak{M}, w \Vdash \varphi$ 不成立;
- $\mathfrak{M}, w \Vdash \varphi \vee \psi$ 当且仅当 $\mathfrak{M}, w \Vdash \varphi$ 或 $\mathfrak{M}, w \Vdash \psi$;
- $\mathfrak{M}, w \Vdash \diamond\varphi$ 当且仅当 存在 $v \in W$, 满足 Rwv 且 $\mathfrak{M}, v \Vdash \varphi$ 。

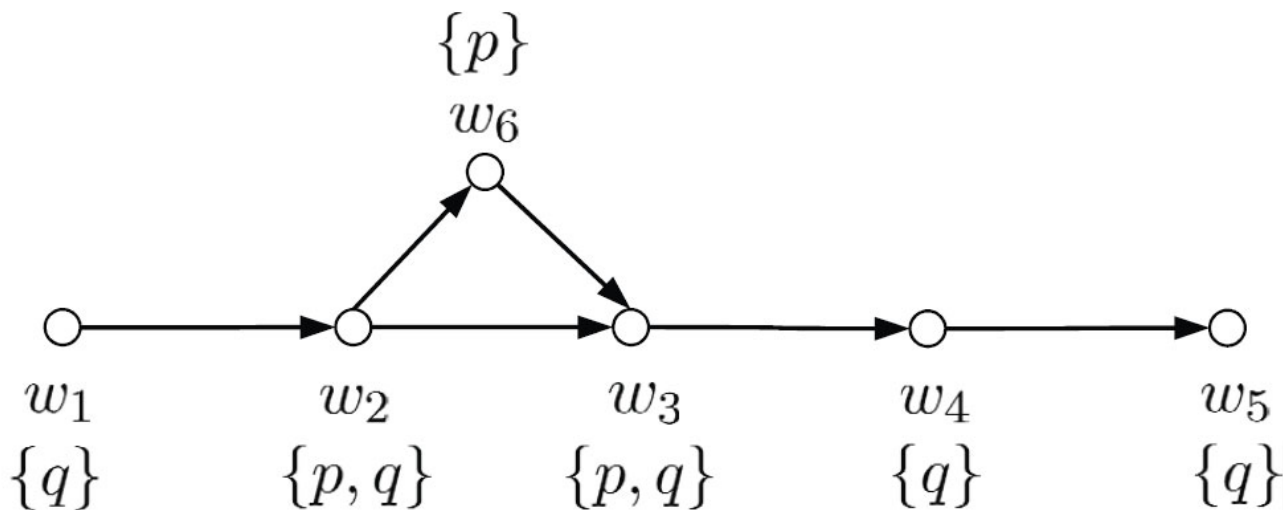
根据上述定义也可以得到：

- $\mathfrak{M}, w \Vdash \square\varphi$ 当且仅当对于任意 $v \in W$, 如果 Rwv 那么 $\mathfrak{M}, v \Vdash \varphi$ 。

可见模态满足性的定义是“内部”和“局部”的。公式的真假是就模型内部的状态而言的，且模态算子 \diamond 的作用是局部的：
它只观察当前状态通过关系 R 能到达的邻居状态。

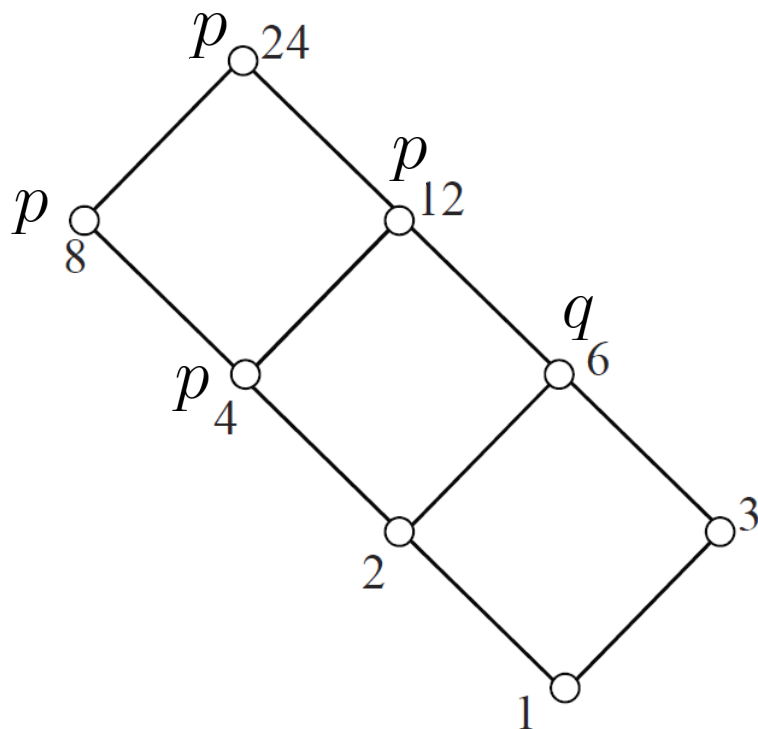


例12.5. (i) 考虑模型如图12.7 所示:



- $\mathfrak{M}, w_1 \Vdash \Diamond \Box p$;
- $\mathfrak{M}, w_1 \nVdash \Diamond \Box p \rightarrow p$;
- $\mathfrak{M}, w_2 \Vdash \Diamond(p \wedge \neg q)$; 以及
- $\mathfrak{M}, w_1 \Vdash q \wedge \Diamond(q \wedge \Diamond(q \wedge \Diamond(q \wedge \Diamond q)))$;
- $\mathfrak{M}, w_5 \Vdash \Box p$ – 空真 (vacuously true)。

(ii) 选用图12.1描述的关系结构作为框架，并定义标记函数如图：



那么可以得到：

- $\mathfrak{M}, 4 \Vdash \Box p$;
- $\mathfrak{M}, 6 \Vdash \Box p$;
- $\mathfrak{M}, 2 \not\Vdash \Box p$; 以及
- $\mathfrak{M}, 2 \Vdash \Diamond (q \wedge \Box p) \wedge \Diamond (\neg q \wedge \Box p)$ 。



模态逻辑的满足关系是定义在模型的状态上的。实际上还可以在框架层次定义一种有效性 (validity)，以使关注点集中于这类框架描述的本体 (ontology) 的特征。

定义12.7. 对于一个任意的模态逻辑公式 φ 。我们称

- φ 在框架 \mathfrak{F} 的状态 w 有效(记作 $\mathfrak{F}, w \Vdash \varphi$)，如果 φ 在任意基于 \mathfrak{F} 的模型 $\mathfrak{M} = (\mathfrak{F}, L)$ 的状态 w 为真；
- φ 在框架 \mathfrak{F} 中有效(记作 $\mathfrak{F} \Vdash \varphi$)，如果它在 \mathfrak{F} 的每个状态上均有效；
- φ 对一类框架 \mathbb{F} 有效(记作 $\mathbb{F} \Vdash \varphi$)，如果它在 \mathbb{F} 中的每个框架中均有效；
- φ 有效(记作 $\Vdash \varphi$)，如果它对于所有类型的框架均是有效的。

对于框架类 \mathbb{F} 有效的所有公式可记作集合 $\Lambda_{\mathbb{F}}$ ，叫作 \mathbb{F} 的逻辑。



命题12.8.(1) 公式 $\diamond(p \vee q) \rightarrow (\diamond p \vee \diamond q)$ 对于所有的框架均有效。

(2) 公式 $\diamond \diamond p \rightarrow \diamond p$ 不是对于所有的框架有效。

(3) 存在一类框架，公式 $\diamond \diamond p \rightarrow \diamond p$ 对这类框架有效。

证明: (1) 欲证这个结论，可以取任意的框架 \mathfrak{F} 以及其中的任意状态 w ，并且令 L 为 \mathfrak{F} 上的一个标注函数，然后证明若 $(\mathfrak{F}, L), w \Vdash \diamond(p \vee q)$ 那么 $(\mathfrak{F}, L), w \Vdash \diamond p \vee \diamond q$ 即可。

假定 $(\mathfrak{F}, L), w \Vdash \diamond(p \vee q)$ 。由定义可知，存在状态 v ，满足 Rwv 且 $(\mathfrak{F}, L), v \Vdash p \vee q$ 。

因此， $v \Vdash p$ 或 $v \Vdash q$ 。

进一步有， $w \Vdash \diamond p$ 或者 $w \Vdash \diamond q$ 。而这两种情况都有 $w \Vdash \diamond p \vee \diamond q$ 。



(2) 欲证这个结论, 我们找出一个框架 \mathfrak{F} , 其中的一个状态 w , 以及一个标记函数 L , 使得上述公式在状态 w 为假即可。

可令 $W = \{0, 1, 2\}$, $R = \{(0, 1), (1, 2)\}$, L 为任意使 $L(2) = \{p\}$ 的标记函数。

那么我们有 $(\mathfrak{F}, L), 0 \Vdash \diamond \diamond p$, 但是 $(\mathfrak{F}, L), 0 \nVdash \diamond p$ 。

因此 $(\mathfrak{F}, L), 0 \nVdash \diamond \diamond p \rightarrow \diamond p$ 。

(3) 可以证明公式 $\diamond \diamond p \rightarrow \diamond p$ 对于传递框架 (transitive frame) 是有效的。所谓传递框架即其中的关系满足传递性的框架。

如果 \mathfrak{F} 是传递框架, 且 w 是其中的任意状态, L 是任意标记函数。

若 $(\mathfrak{F}, L), w \Vdash \diamond \diamond p$, 那么由定义, 有状态 u 和 v , Rwu , Ruv 并且 $(\mathfrak{F}, L), v \Vdash p$ 。

但是由于 R 是传递的, 我们可得到 Rwv , 因此有 $(\mathfrak{F}, L), w \Vdash \diamond p$ 。

进而有 $(\mathfrak{F}, L), w \Vdash \diamond \diamond p \rightarrow \diamond p$ 。 □



线性时态逻辑

下面介绍的时态语言主要来自于20世纪七八十年代自动形式化验证领域的研究成果。

定义12.9. 线性时态逻辑的模型为线性时间模型 $\mathfrak{M} = (S, x, L)$, 其中

- S 是一个非空状态集;
- $x : \mathbb{N} \rightarrow S$ 是一个状态的无穷序列;
- $L : W \rightarrow 2^\Phi$ 为标记函数, 把 W 中的各个点标记上在该点为真的命题符。

其中 Φ 是一个潜在的命题符的集合。



定义12.3. 线性时间时态语言(Linear-time Temporal Language) 基于一个命题符的集合 Φ 以及线性时间时态算子 \mathcal{U} (“Until”) 和 \mathcal{O} (“neXt-time”) 而定义, 其合式公式 ψ 由以下规则给出:

$$\psi ::= p \mid \perp \mid \neg\psi \mid \psi_1 \vee \psi_2 \mid \mathcal{O}\psi \mid \psi_1\mathcal{U}\psi_2$$

其中 $p \in \Phi$ 。通常假设 Φ 是一个可数无穷集 $\{p_0, p_1, \dots\}$, 在某些情况下也能假定其为一个有穷集或不可数无穷集。



此外，还可以定义一些常用的时态算子如下：

- (“Finally”) $\diamond\psi := \top\mathcal{U}\psi$
- (“Globally”) $\square\psi := \neg\diamond\neg\psi$
- (“Infinitely Often”) $\overset{\infty}{\diamond}\psi := \square\diamond\psi$
- (“Almost Everywhere”) $\overset{\infty}{\square}\psi := \diamond\square\psi$
- (“Release”) $\psi_1\mathcal{R}\psi_2 := \neg(\neg\psi_1\mathcal{U}\neg\psi_2)$

在一些文献中，时态算子 \bigcirc 、 \diamond 、 \square 、 \mathcal{U} 、 \mathcal{R} 也分别被记作 X、F、G、U、R。



定义12.10 可基于线性时间模型 $\mathfrak{M} = (S, x, L)$ 定义线性时间时态逻辑的语义。 $\mathfrak{M}, x \models \psi$ 表示“在模型 \mathfrak{M} 的时间线 x 上公式 ψ 为真”。 满足关系 \models 可归纳地定义如下：

- $\mathfrak{M}, x \models p$ 当且仅当 $p \in L(s_0)$, 其中 $p \in \Phi$;
- $\mathfrak{M}, x \models \perp$ 从不成立;
- $\mathfrak{M}, x \models \neg\psi$ 当且仅当 $\mathfrak{M}, x \models \psi$ 不成立;
- $\mathfrak{M}, x \models \psi_1 \vee \psi_2$ 当且仅当 $\mathfrak{M}, x \models \psi_1$ 或 $\mathfrak{M}, x \models \psi_2$;
- $\mathfrak{M}, x \models \psi_1 \mathcal{U} \psi_2$ 当且仅当 $\exists j(\mathfrak{M}, x^j \models \psi_2$ 以及 $\forall k < j(\mathfrak{M}, x^k \models \psi_1)$);
- $\mathfrak{M}, x \models \bigcirc\psi$ 当且仅当 $\mathfrak{M}, x^1 \models \psi$ 。

其中 x^i 表示路径 x 的后缀 $s_i, s_{i+1}, s_{i+2}, \dots$ 。



例12.6. LTL的一些公式对应的满足时间线的模式如图12.8所示:

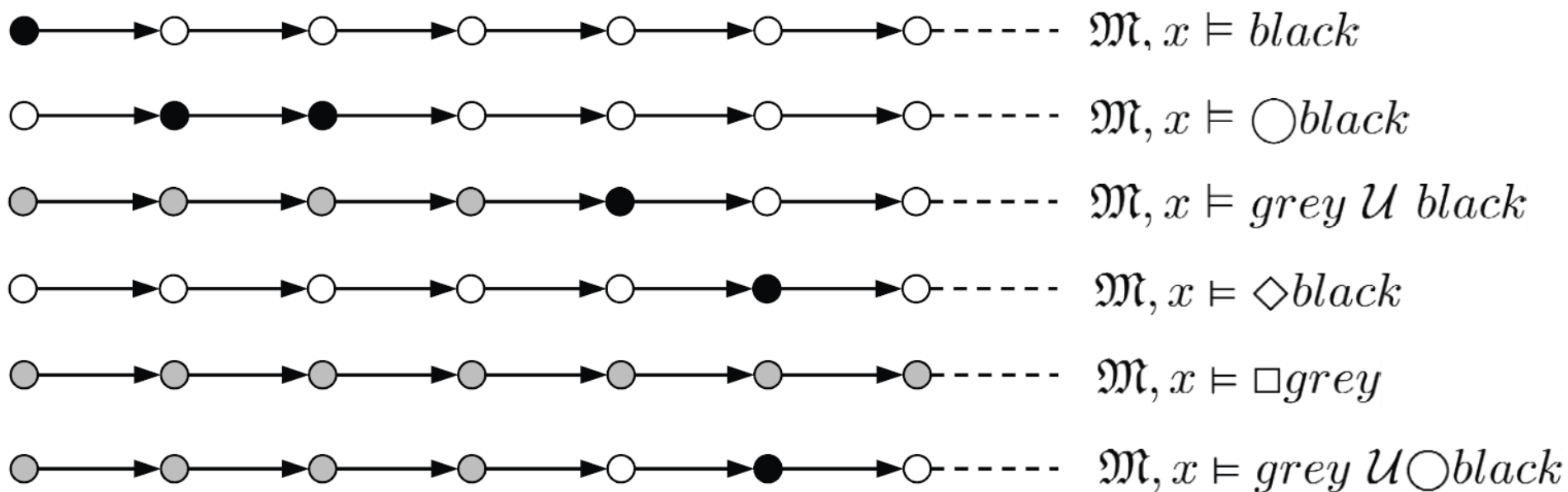


图 12.8 : 线性时间时态逻辑语义示例



分支时态逻辑（计算树逻辑）

用于为分支时态逻辑提供语义解释的数学结构是如下一个 Kripke 模型，我们把它叫做分支时间模型。

在有的文献中也把它叫做转换系统。

定义12.11. 分支时间模型模型为 $\mathfrak{M} = (S, R, L)$ 其中

- S 是一个非空状态集;
- $R \subseteq S \times S$ 是一个完全的二元关系（即 $\forall s \in S \exists t \in S : (s, t) \in R$ ）;
- $L : S \rightarrow 2^\Phi$ 为标记函数，把 S 中的各个点标记上为真的命题符。

其中 Φ 是一个潜在的命题符的集合。

可见分支时间模型实际上是由一个分支时间结构 $\mathfrak{F} = (S, R)$ 以及一个标记函数 L 构成的。



定义12.4. 分支时间时态语言 (Branching-time Temporal Language) 由一个命题符的集合 Φ 、线性时态算子、以及路径选择算子 \exists (“for some futures”) 生成。可定义两类公式，分别是路径公式 (path formula) ψ 和状态公式 (state formula) φ ，它的合式公式分别由以下规则给出：

$$\begin{aligned}\varphi &::= p \mid \perp \mid \varphi_1 \vee \varphi_2 \mid \neg\varphi \mid \exists\psi \\ \psi &::= \varphi \mid \psi_1 \vee \psi_2 \mid \neg\psi \mid \bigcirc\psi \mid \psi_1 \mathcal{U}\psi_2\end{aligned}$$

上述规则生成的状态公式构成了分支时间时态语言。

此外还可以定义另一个常用的路径选择算子 \forall (“for all futures”) 为： $\forall\psi := \neg\exists\neg\psi$ 。



此外还可以上述时态语言的一种子语言 (sublanguage) 为: 状态公式 φ 的定义不变, 而路径公式 ψ 的规则变为:

$$\psi ::= \bigcirc\varphi \mid \varphi_1\mathcal{U}\varphi_2$$

即限制原路径公式语法, 不允许线性时态算子的布尔组合和嵌套。容易发现, 它实际上等价于以下语法规则:

$$\varphi ::= p \mid \perp \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \exists\bigcirc\varphi \mid \exists\Box\varphi \mid \exists(\varphi_1\mathcal{U}\varphi_2)$$

上述语言中, 可以把 $\exists\bigcirc$ 、 $\exists\Box$ 和 $\exists\mathcal{U}$ 看作基本的时态算子。此外还可以基于此定义其它5个时态算子如下:

- $\forall\bigcirc\varphi := \neg\exists\bigcirc\neg\varphi$ $\forall\Box\varphi := \neg\exists\Diamond\neg\varphi$
- $\forall\Diamond\varphi := \neg\exists\Box\neg\varphi$ $\exists\Diamond\varphi := \exists(\top\mathcal{U}\varphi)$
- $\forall(\varphi_1\mathcal{U}\varphi_2) := \neg\exists(\neg\varphi_2\mathcal{U}\neg\varphi_1 \wedge \neg\varphi_2) \wedge \neg\exists\Box\neg\varphi_2$



进而我们可以定义计算树逻辑（Computation Tree Logic），
简称为CTL*（简化版本简称为CTL）的语义。

定义12.12. 对于模型 $\mathfrak{M} = (S, R, L)$, 无穷状态序列 $x = (s_0, s_1, \dots)$ 是一条全路径（fullpath）当且仅当 $\forall i \in \mathbb{N} : (s_i, s_{i+1}) \in R$ 。
对于 CTL* 的任意状态公式 φ 和路径公式 ψ ,

- $\mathfrak{M}, s_0 \models \varphi$ 表示 φ 在 \mathfrak{M} 的状态 s_0 为真；
- $\mathfrak{M}, x \models \psi$ 表示 ψ 对于 \mathfrak{M} 中全路径 x 为真。

关系 \models 可归纳地定义如下：



(S1) $\mathfrak{M}, s_0 \models p$ 当且仅当 $p \in L(s_0)$;
 $\mathfrak{M}, s_0 \models \perp$ 从不成立;

(S2) $\mathfrak{M}, s_0 \models \varphi_1 \vee \varphi_2$ 当且仅当 $\mathfrak{M}, s_0 \models \varphi_1$ 或 $\mathfrak{M}, s_0 \models \varphi_2$;
 $\mathfrak{M}, s_0 \models \neg\varphi$ 当且仅当 $\mathfrak{M}, s_0 \models \varphi$ 不成立;

(S3) $\mathfrak{M}, s_0 \models \exists\psi$ 当且仅当 \mathfrak{M} 中存在全路径 $x = (s_0, s_1, \dots)$,
满足 $\mathfrak{M}, x \models \psi$;

(P1) $\mathfrak{M}, x \models \varphi$ 当且仅当 $\mathfrak{M}, s_0 \models \varphi$;

(P2) $\mathfrak{M}, x \models \psi_1 \vee \psi_2$ 当且仅当 $\mathfrak{M}, x \models \psi_1$ 或 $\mathfrak{M}, x \models \psi_2$;
 $\mathfrak{M}, x \models \neg\psi$ 当且仅当 $\mathfrak{M}, x \models \psi$ 不成立;

(P3) $\mathfrak{M}, x \models \psi_1 \mathcal{U} \psi_2$ 当且仅当
 $\exists j (\mathfrak{M}, x^j \models \psi_2$ 以及 $\forall k < j (\mathfrak{M}, x^k \models \psi_1))$;
 $\mathfrak{M}, x \models \bigcirc\psi$ 当且仅当 $\mathfrak{M}, x^1 \models \psi$ 。



CTL 作为 CTL* 的子集，上述语义定义自然也完全适用。
但是可以采用更为简洁的语义定义，具体而言包括上面的
S1, S2, S3以及下面的S4:

- (S4) $\mathfrak{M}, s_0 \models \exists \bigcirc \varphi$ 当且仅当 \mathfrak{M} 中存在状态 s_1 满足 Rs_0s_1 ,
且 $\mathfrak{M}, s_1 \models \varphi$;
 $\mathfrak{M}, s_0 \models \exists \square \varphi$ 当且仅当 \mathfrak{M} 中存在全路径 $x = (s_0, s_1, \dots)$,
满足 $\forall i \in \mathbb{N} : \mathfrak{M}, s_i \models \varphi$;
 $\mathfrak{M}, s_0 \models \exists (\varphi_1 \mathcal{U} \varphi_2)$ 当且仅当 \mathfrak{M} 中存在全路径 $x = (s_0, s_1, \dots)$,
满足 $\exists j (\mathfrak{M}, s_j \models \varphi_2$ 以及 $\forall k < j (\mathfrak{M}, s_k \models \varphi_1))$;

例12.7. 对于如图12.9所示的模型，
我们可以得到：

(1) $\mathfrak{M}, d \models \text{black};$

(2) $\mathfrak{M}, a \models \forall \bigcirc \text{grey};$

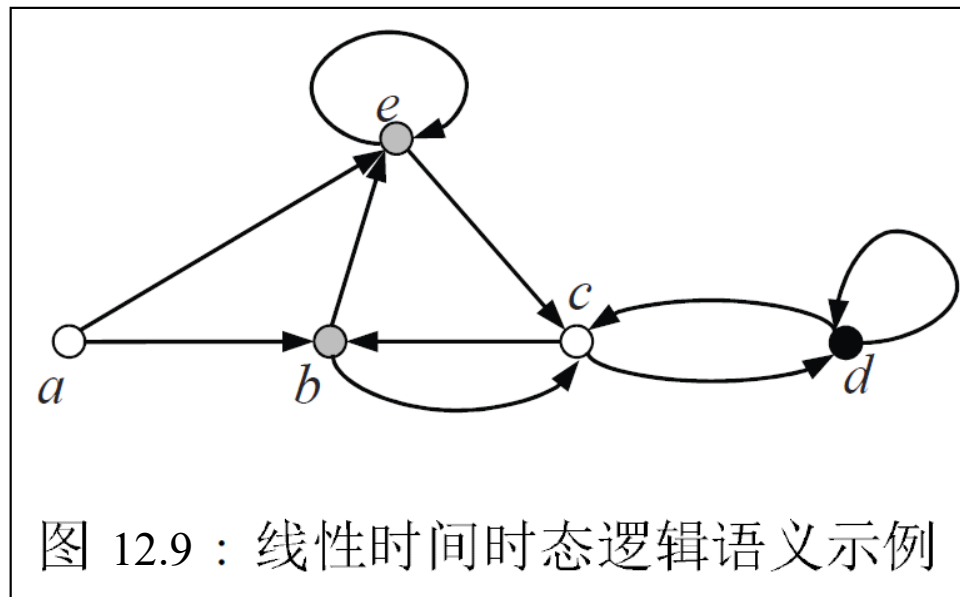
(3) $\mathfrak{M}, a \models \exists \diamond \exists \square \text{black};$

(4) $\mathfrak{M}, b \models \exists(\text{grey } \mathcal{U} \text{white});$

(5) $\mathfrak{M}, e \models \forall(\text{grey } \mathcal{U} (\text{white} \wedge \exists \bigcirc \exists \square \text{black}));$

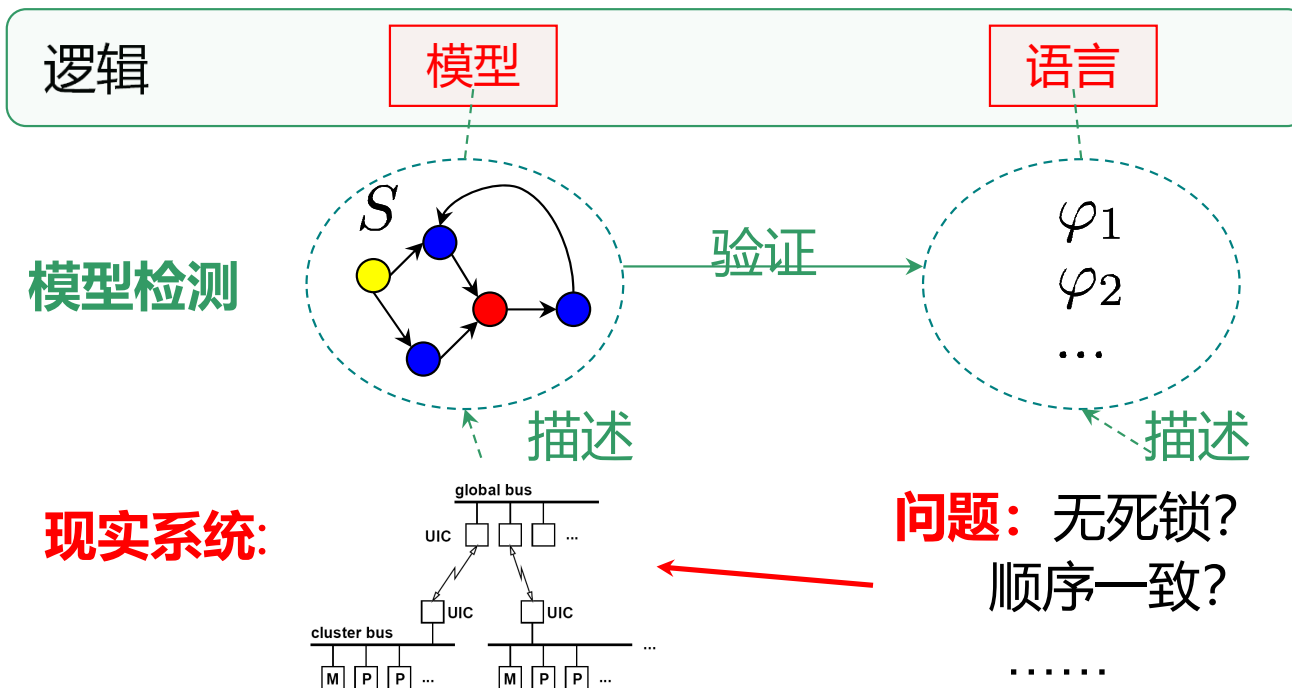
(6) $\mathfrak{M}, c \models \forall(\bigcirc \text{grey} \vee \bigcirc \text{black});$

(7) $\mathfrak{M}, a \models \exists(\text{white} \wedge \bigcirc \text{grey} \wedge \bigcirc \bigcirc \text{grey} \wedge \diamond \exists \square \text{black}).$





例12.8. CTL在用于计算机软硬件系统自动验证的模型检测技术中得到了成功的应用。



- $\exists \diamond (Started \wedge \neg Ready)$: 到达一个已启动但并未就绪的状态是可能的;
- $\forall \square (Req \rightarrow \forall \diamond Ack)$: 如果发生请求那么会被确认收到;
- $\forall \square (\forall \diamond DeviceEnabled)$: 一个设备总是可用的;
- $\forall \square (\exists \diamond Restart)$: 重启总是可能的。



模态推理系统

- 模态逻辑其实包括许多适用于不同框架、模型，采用不同语言的逻辑系统。如何为这些逻辑构建公理系统，从而由语法机制生成所有的在我们关注的框架类上有效的公式？
- 一种思路是首先构建一种最一般、最基本的公理系统，然后对于各种不同的逻辑可以继承上述基本系统，并添加相应的特征性公理，从而构成适用于这种逻辑的更强的系统。
- 我们将定义一种基本模态语言的Hilbert公理系统 \mathbf{K} 。可证 \mathbf{K} 是上述基本系统，它实际上是用于就框架推理的“最小” (或“最弱”)的系统，更强的系统可通过添加额外公理得到。



定义12.13. **K-证明**是一个有穷的公式序列，其中任何一个公式或者是公理，或者是由序列中排在前面的一个或多个公式通过采用一条或多条规则得到。

K-系统的公理包括以下三部分：

- **(TAUT)** 所有的重言式;
- **(K)** $\Box(p \rightarrow q) \rightarrow (\Box p \rightarrow \Box q)$;
- **(Dual)** $\Diamond p \leftrightarrow \neg \Box \neg p$ 。



K-系统的规则包括:

- (Modus Ponens, **MP**) $\frac{\varphi \rightarrow \psi, \varphi}{\psi}$;
- (Uniform substitution, **US**) $\frac{\varphi}{\theta}$ 其中 θ 为把 φ 中的命题符一致地替换为任意的公式后得到的公式;
- (Generalization, **N**) $\frac{\varphi}{\Box\varphi}$ 。

如果一个公式 φ 出现为某个**K**-证明的最后一个公式, 那么我们就说 φ 是**K**-可证的, 并记作 $\vdash_{\mathbf{K}} \varphi$ 。



K-系统在如下意义下是最小模态Hilbert系统:

- 很容易证明**K**-系统的公理均是有效的，而且**K**-系统的三条规则保持有效性，因此所有的**K**-可证的公式均是有效的。
即**K**-系统对于所有的框架构成的类是可靠的（sound）。
- 可证反过来也是正确的：如果一个基本模态公式是有效的，那么它就是**K**-可证的。也就是说对于所有的框架构成的类是完全的（complete）。

简而言之，**K**-系统恰好产生所有的基本模态逻辑有效公式。

定义12.14. **K**-系统对于所有的框架是可靠且完全的。



例12.9. 公式 $(\Box p \wedge \Box q) \rightarrow \Box(p \wedge q)$ 对于任何框架均是有效的, 因而它应该是**K**-可证的。下面证明过程说明事实上的确如此:

- | | |
|---|-----------------|
| (1) $\vdash p \rightarrow (q \rightarrow (p \wedge q))$ | TAUT/A09 |
| (2) $\vdash \Box(p \rightarrow (q \rightarrow (p \wedge q)))$ | N(1) |
| (3) $\vdash \Box(p \rightarrow q) \rightarrow (\Box p \rightarrow \Box q)$ | K |
| (4) $\vdash \Box(p \rightarrow (q \rightarrow (p \wedge q))) \rightarrow (\Box p \rightarrow \Box(q \rightarrow (p \wedge q)))$ | US(3) |
| (5) $\vdash \Box p \rightarrow \Box(q \rightarrow (p \wedge q))$ | MP(2)(4) |
| (6) $\vdash \Box(q \rightarrow (p \wedge q)) \rightarrow (\Box q \rightarrow \Box(p \wedge q))$ | US(3) |
| (7) $(5) \rightarrow ((6) \rightarrow (8))$ | TAUT/A03 |
| (8) $\vdash \Box p \rightarrow (\Box q \rightarrow \Box(p \wedge q))$ | MP(MP(7)(5))(6) |
| (9) $(8) \rightarrow (10)$ | TAUT/T28 |
| (10) $\vdash (\Box p \wedge \Box q) \rightarrow \Box(p \wedge q)$ | MP(9)(8) |



我们可以为**K**系统添加公理 $\diamond \diamond p \rightarrow \diamond p$ ，从而获得一个叫做**K4**的Hilbert系统。可以证明 **K4** 对于所有的传递框架是可靠和完全的 (即恰好生成所有的在传递框架上有效的公式),并且可以证明对于任意公式集 Σ 以及公式 φ :

$$\Sigma \vdash_{K4} \varphi \quad \text{iff} \quad \Sigma \Vdash_{Tran} \varphi,$$

其中 $\Sigma \vdash_{K4} \varphi$ 当且仅当存在 Σ 的一个有穷子集 $\{\sigma_1, \dots, \sigma_n\}$ 使得 $\vdash_{K4} \sigma_1 \wedge \dots \wedge \sigma_n \rightarrow \varphi$ 。而 \Vdash_{Tran} 表示传递框架上的局部语义后承。

简而言之，我们把传递框架上的局部语义后承关系化归到 **K4** 上的可证明性。



更一般地，可以把基本模态逻辑的任意公式集 Γ 作为新的公理加入到 **K** 系统中,从而构成公理系统 **K** Γ ,很多情况下都可以得到类似的框架有效性结论。

所有这类公理系统各自能生成的公式集都可以被纳入到正规模态逻辑的概念下。

定义12.15. 一个正规模态逻辑 Λ 是如下一个公式集:

- (i) 包含所有的重言式、以及 $\Box(p \rightarrow q) \rightarrow (\Box p \rightarrow \Box q)$ 和 $\Diamond p \leftrightarrow \neg \Box \neg p$;
- (ii) 对规则 **MP**, **US** 和 **N** 封闭。

我们把最小的一个正规模态逻辑叫做 **K**。



上述定义直接抽象于模态Hilbert系统的潜在思想。它抛弃所有的关于证明顺序的讨论并专注于真正本质性的部分：存在公理，并且对证明规则封闭。

可以证明：对于任意框架类 \mathbb{F} , 所有在其上有效的公式构成的集合 $\Lambda_{\mathbb{F}}$ 是一个正规模态逻辑。

也就是说正规模态逻辑的概念也能很好地对应到语义层面。



K: $\Box(p \rightarrow q) \rightarrow (\Box p \rightarrow \Box q)$ (Distribution Axiom)

T: $\Box p \rightarrow p$ (Reflexivity Axiom)

4: $\Box p \rightarrow \Box \Box p$ (or $\Diamond \Diamond p \rightarrow \Diamond p$)

B: $p \rightarrow \Box \Diamond p$

D: $\Box p \rightarrow \Diamond p$

5: $\Diamond p \rightarrow \Box \Diamond p$

$K := \mathbf{K} + \mathbf{N} + \mathbf{US} + \mathbf{MP}$

$T := K + \mathbf{T}$

$S4 := T + \mathbf{4}$

$S5 := S4 + \mathbf{5}$

$D := K + \mathbf{D}$



从模态逻辑到一阶逻辑

为了说明模态逻辑并不是一个孤立的形式化系统，我们可以在模态逻辑和一阶逻辑之间架起一座桥梁。为获得简洁的表述，我们把关注点放在基本模态逻辑。

定义12.16. 对于一个命题符的集合 Φ , $\mathcal{L}^1(\Phi)$ 为如下带等词的一阶语言：

- (i) 具有一元谓词 P_0, P_1, P_2, \dots 分别对应于 Φ 中的命题符 p_0, p_1, p_2, \dots ;
- (ii) 具有一个二元关系 R , 对应于模态算子 \diamond 。



定义12.17. 令 x 为一阶逻辑的变元。把基本模态语言公式对应到 $\mathcal{L}^1(\Phi)$ 中的一阶语言公式的标准翻译 ST_x 归纳地定义如下：

- $ST_x(p) = Px$;
- $ST_x(\perp) = x \neq x$;
- $ST_x(\neg\phi) = \neg ST_x(\phi)$;
- $ST_x(\phi \vee \psi) = ST_x(\phi) \vee ST_x(\psi)$;
- $ST_x(\diamond\phi) = \exists y(Rxy \wedge ST_y(\phi))$ 。

其中 y 是新变元。



例12.10. $\Box\varphi$ 和 $\Diamond(\Box p \rightarrow q)$ 的标准翻译分别如下:

$$\begin{aligned}(1) \quad ST_x(\Box\varphi) &= ST_x(\neg \Diamond \neg\varphi) \\ &= \neg \exists y (Rxy \wedge ST_y(\neg\varphi)) \\ &= \forall y (Rxy \rightarrow ST_y(\varphi));\end{aligned}$$

$$\begin{aligned}(2) \quad ST_x(\Diamond(\Box p \rightarrow q)) &= \exists y_1 (Rxy_1 \wedge ST_{y_1}(\Box p \rightarrow q)) \\ &= \exists y_1 (Rxy_1 \wedge (ST_{y_1}(\Box p) \rightarrow ST_{y_1}(q))) \\ &= \exists y_1 (Rxy_1 \wedge (\forall y_2 (Ry_1y_2 \rightarrow ST_{y_2}(p)) \rightarrow Qy_1)) \\ &= \exists y_1 (Rxy_1 \wedge (\forall y_2 (Ry_1y_2 \rightarrow Py_2) \rightarrow Qy_1))\end{aligned}$$



标准翻译的合理性显而易见：

它实质上把模态满足的定义用一阶语言重新描述。

- $ST_x(\varphi)$ 将包含恰好一个自由变元 x , 其作用实际上是用于标注当前状态, 使得一阶逻辑的全局观念能够模拟模态满足的局部观念。
- 模态词被翻译为受限的量词, 即被限制为仅作用于相关的状态, 这显然是一种用一阶逻辑模拟模态词的局部作用的方法。
- 此外, 基于 Φ 的基本模态语言的模型也可以被看作 $\mathcal{L}^1(\Phi)$ 的模型。



我们用 $\mathfrak{M} \models ST_x(\varphi)[w]$ 来表示当 w 被赋值给自由变元 x 时，一阶语言公式 $ST_x(\varphi)$ 在模型 \mathfrak{M} 被满足。

定理12.18. 若 φ 是一个基本模态语言公式， \mathfrak{M} 是一个任意的模型， w 是其上的一个任意状态，那么

- (i) $\mathfrak{M}, w \Vdash \varphi$ 当且仅当 $\mathfrak{M} \models ST_x(\varphi)[w]$;
- (ii) $\forall w : \mathfrak{M}, w \Vdash \varphi$ 当且仅当 $\mathfrak{M} \models \forall x ST_x(\varphi)$ 。

证明: 可通过对 φ 的结构进行归纳证明。具体过程留作习题。□

上次结论说明，当在模型层次进行解释时，基本模态语言公式等价于具有一个自由变元的一阶语言公式。



本讲小结

- **Slogan 1:** Modal languages are simple yet expressive language for talking about relation structures;
- **Slogan 2:** Modal languages provide an internal, local perspective on relation structures;
- **Slogan 3:** Modal languages are not isolated formal systems.



Recommended Books

- [1] Patrick Blackburn, Maarten de Rijke, Yde Venema, *Modal Logic*, Cambridge University Press, 2001.
- [2] E. Allen Emerson, Temporal and Modal Logic, In *handbook of Theoretical Computer Science*, 1990.
- [3] Brian F. Chellas, *Modal Logic: A Introduction*, Cambridge University Press, 1980.
- [4] Ronald Fagin, Joseph Y. Halpern, Yoram Moses, Moshe Y. Vardi, *Reasoning about Knowledge*, The MIT Press, 1995.



The End of Lecture 12