形式化方法导引

第 5 章 模型检测

5.1 应用

黄文超
https://faculty.ustc.edu.cn/huangwenchao
$\longrightarrow$ 教学课程 $\longrightarrow$ 形式化方法导引

回顾: 定义: Verification in Logics

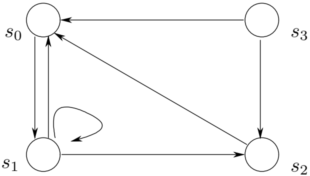
Most logics used in the design, specification and verification of computer systems fundamentally deal with a *satisfaction relation*:

$$\mathcal{M} \models \phi$$

- $\mathcal{M}$ is some sort of situation or *model* of a system
- ϕ is a *specification*, a formula of that logic, expressing what should be true in situation $\mathcal{M}$.
- At the *heart* of this set-up is that one can often *specify and implement algorithms* for computing $\models$.

回顾: 下一个问题:

- 问: 如何统一化定义 $\mathcal{M}$ 和 ϕ? 答: 一种方案: $\mathcal{M}$ 和 ϕ 均用 *Logics*
  - Propositional logic, First-order logic, Higher-order logic



反例：How to define Reachability as ϕ

Given nodes n and n' in a directed graph, is there a finite path of transitions from n to n'?

反例：一种答案

$$(u = v) \vee \exists x(R(u,x) \wedge R(x,v)) \vee \exists x_1 \exists x_2 (R(u,x_1) \wedge R(x_1,x_2) \wedge R(x_2,v)) \vee \ldots$$

- This is infinite, so it’s *not* a well-formed formula.
- Can we find a well-formed formula with the same meaning? *No!*

回顾：另一种答案：Second-order Logic

$$\neg\exists P \forall x \forall y \forall z\ (C_1 \wedge C_2 \wedge C_3 \wedge C_4)$$

where

$$C_1 \stackrel{\text{def}}{=} P(x,x)$$
$$C_2 \stackrel{\text{def}}{=} P(x,y) \wedge P(y,z) \rightarrow P(x,z)$$
$$C_3 \stackrel{\text{def}}{=} P(u,v) \rightarrow \bot$$
$$C_4 \stackrel{\text{def}}{=} R(x,y) \rightarrow P(x,y)$$

问题:

- 难以**理解**ϕ，难以**构建**ϕ
- 如何**自动**验证?

回顾: 下一个问题:

- 问: 如何统一化定义 $\mathcal{M}$ 和 ϕ? 答: 一种方案: $\mathcal{M}$ 和 ϕ 均用 *Logics*
    - Propositional logic, First-order logic, Higher-order logic

还有一个大问题:

用 Logics 来直接构建 $\mathcal{M}$ 的缺点?

- 不够直观
- 怎样自动化?

本章内容: (本节-1. 应用)

- 模型检测 (重新定义问题):
  - 重新定义ϕ: LTL, CTL, ...
  - 重新定义$\mathcal{M}$: Transition System 等
- NuSMV 语言的使用

下一节 (预告) (2. 理论)

- 如何设计算法求解上述问题
  - 利用 SAT 求解工具, 如 BMC
  - 设计新的算法, 如 BDD

1. 应用

1.1. Model Checking | 定义

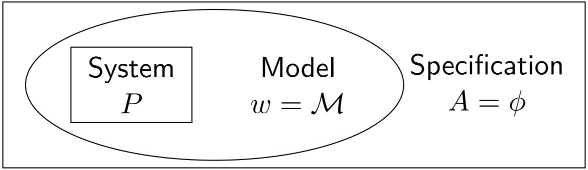
再往前回顾: 定义: Verifier

A *verifier* for a language A is an algorithm V, where

$$A = \{w \mid V \text{ accepts } \langle w, c\rangle \text{ for some string } c\}.$$

再往前回顾: 验证过程

(1) 构建**模型** w. **(2)** 设计**规约** A. **(3)** (手动或自动) 构建**证明** c
(4) 使用验证器 V, 输入 c, 输出是否$w \in A$



Model Checking: (1) $\mathcal{M} \Rightarrow \mathcal{M}, s$ (2) ϕ: classical logic $\Rightarrow$ temporal logic
详见后页

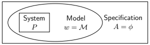
再往前回顾: 定义: Verifier

A *verifier* for a language A is an algorithm V, where

$$A = \{w \mid V \text{ accepts } \langle w, c\rangle \text{ for some string } c\}.$$

再往前回顾: 验证过程

(1) 构建模型 w. **(2)** 设计规约 A. **(3)** (手动或自动) 构建证明 c
(4) 使用验证器 V, 输入 c, 输出是否 $w \in A$



Model Checking: (1) $\mathcal{M} \Rightarrow \mathcal{M}, s$ (2) ϕ: classical logic $\Rightarrow$ temporal logic
详见后页

To verify that a system satisfies a property, we must do three things:

- model the system using the description language of a model checker, arriving at a model $\mathcal{M}$;
- code the property using the specification language of the model checker, resulting in a temporal logic formula ϕ;
- Run the model checker with inputs $\mathcal{M}$ and ϕ.

Models like $\mathcal{M}$ should not be confused with an *actual physical system*. Models are abstractions that omit lots of real features of a physical system, which are irrelevant to the checking of ϕ. This is similar to the abstractions that one does in calculus or mechanics. There we talk about straight lines, perfect circles, or an experiment without friction. These abstractions are very powerful, for they allow us to focus on the essentials of our particular concern.

定义: Model checking

Model checking is the process of computing an answer to the question of whether $\mathcal{M}, s \vDash \phi$ holds, where

- $\mathcal{M}$ is an *appropriate* model of the system under consideration.
- s is a *state* of that model
- $\vDash$ is the underlying satisfaction relation
- ϕ is a formula of one of the following *temporal logics*:
  - *Linear-time Temporal Logic (LTL)*
  - *Computation Tree Logic (CTL)*
  - etc.

下一个问题: temporal logic? LTL? CTL?

定义: Linear-time temporal logic (LTL)

Linear-time temporal logic (*LTL*) has following syntax given in BNF:

$$\begin{aligned}\phi ::= & \top \mid \bot \mid p \mid (\neg\phi) \mid (\phi \wedge \phi) \mid (\phi \vee \phi) \mid (\phi \rightarrow \phi) \\ & \mid (\mathrm{X}\ \phi) \mid (\mathrm{F}\ \phi) \mid (\mathrm{G}\ \phi) \mid (\phi\ \mathrm{U}\ \phi) \mid (\phi\ \mathrm{W}\ \phi) \mid (\phi\ \mathrm{R}\ \phi)\end{aligned}$$

where p is any propositional atom from some set **Atoms**.

Convention: The unary connectives (consisting of $\neg$ and the temporal connectives X, F and G) bind most tightly. Next in the order come U, R and W; then come $\wedge$ and $\vee$; and after that comes $\rightarrow$. For example:

- $(((\mathrm{F}\ p) \wedge (\mathrm{G}\ q)) \rightarrow (p\ \mathrm{W}\ r)) \equiv \mathrm{F}\ p \wedge \mathrm{G}\ q \rightarrow p\ \mathrm{W}\ r$
- $(\mathrm{F}\ (p \rightarrow (\mathrm{G}\ r)) \vee ((\neg q)\ \mathrm{U}\ p)) \equiv \mathrm{F}\ (p \rightarrow \mathrm{G}\ r) \vee \neg q\ \mathrm{U}\ p$
- $(p\ \mathrm{W}\ (q\ \mathrm{W}\ r)) \equiv p\ \mathrm{W}\ (q\ \mathrm{W}\ r)$
- $((\mathrm{G}\ (\mathrm{F}\ p)) \rightarrow (\mathrm{F}\ (q \vee s))) \equiv \mathrm{G}\ \mathrm{F}\ p \rightarrow \mathrm{F}\ (q \vee s)$

1. 应用

1.2 Linear-time temporal Logic (LTL)

定义: Linear-time temporal logic (LTL)

Linear-time temporal logic (*LTL*) has following syntax given in BNF:

$$\begin{aligned}\phi ::= & \top \mid \bot \mid p \mid (\neg\phi) \mid (\phi \wedge \phi) \mid (\phi \vee \phi) \mid (\phi \to \phi) \\ & \mid (\mathrm{X}\ \phi) \mid (\mathrm{F}\ \phi) \mid (\mathrm{G}\ \phi) \mid (\phi\ \mathrm{U}\ \phi) \mid (\phi\ \mathrm{W}\ \phi) \mid (\phi\ \mathrm{R}\ \phi)\end{aligned}$$

where p is any propositional atom from some set **Atoms**.

Convention: The unary connectives (consisting of $\neg$ and the temporal connectives X, F and G) bind most tightly. Next in the order come U, R and W; then come $\wedge$ and $\vee$; and after that comes $\to$. For example:

- $(((\mathrm{F}\ p) \wedge (\mathrm{G}\ q)) \to (p\ \mathrm{W}\ r)) \equiv \mathrm{F}\ p \wedge \mathrm{G}\ q \to p\ \mathrm{W}\ r$
- $(\mathrm{F}\ (p \to (\mathrm{G}\ r)) \vee ((\neg q)\ \mathrm{U}\ p)) \equiv \mathrm{F}\ (p \to \mathrm{G}\ r) \vee \neg q\ \mathrm{U}\ p$
- $(p\ \mathrm{W}\ (q\ \mathrm{W}\ r)) \equiv p\ \mathrm{W}\ (q\ \mathrm{W}\ r)$
- $((\mathrm{G}\ (\mathrm{F}\ p)) \to (\mathrm{F}\ (q \vee s))) \equiv \mathrm{G}\ \mathrm{F}\ p \to \mathrm{F}\ (q \vee s)$

It’s boring to write all those brackets, and makes the formulas hard to read. Many of them can be omitted without introducing ambiguities; for example, $(p \to (\mathrm{F}\ q))$ could be written $p \to \mathrm{F}\ q$ without ambiguity. Others, however, are required to resolve ambiguities. In order to omit some of those, we assume similar binding priorities for the LTL connectives to those we assumed for propositional and predicate logic.

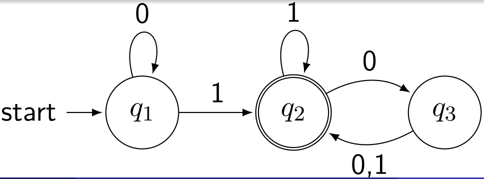
问题: 定义 (理解) 新的符号? 如: X, F, G, U, W, R

思路: 先定义一种**最简化的模型**, 然后利用这个模型来定义符号

回顾: 定义: finite automaton

A *finite automaton* is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

1. Q is a finite set called the *states*,
2. Σ is a finite set called the *alphabet*,
3. $\delta : Q \times \Sigma \to Q$ is the *transition function*,
4. $q_0 \in Q$ is the *start state*, and
5. $F \subseteq Q$ is the set of *accept states*.

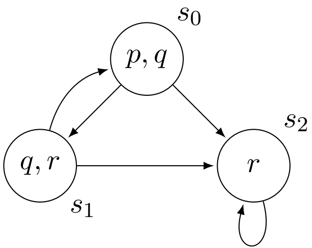


问题: 定义 (理解) 新的符号? 如: X, F, G, U, W, R
思路: 先定义一种最简化的模型, 然后利用这个模型来定义新符号

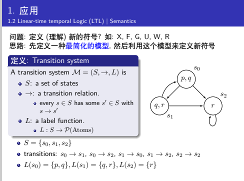
定义: Transition system

A transition system $\mathcal{M} = (S, \to, L)$ is

- S: a set of states
- $\to$: a transition relation.
  - every $s \in S$ has some $s' \in S$ with $s \to s'$
- L: a label function.
  - $L : S \to \mathcal{P}(\mathrm{Atoms})$



- $S = \{s_0, s_1, s_2\}$
- transitions: $s_0 \to s_1$, $s_0 \to s_2$, $s_1 \to s_0$, $s_1 \to s_2$, $s_2 \to s_2$
- $L(s_0) = \{p, q\}, L(s_1) = \{q, r\}, L(s_2) = \{r\}$



Transition systems are also simply called models in this chapter. So a model has a collection of states S, a relation $\rightarrow$, saying how the system can move from state to state, and, associated with each state s, one has the set of atomic propositions $L(s)$ which are true at that particular state. We write $\mathcal{P}(\mathrm{Atoms})$ for the *power set* of Atoms, a collection of atomic descriptions. For example, the power set of $\{p, q\}$ is $\{\emptyset, \{p\}, \{q\}, \{p, q\}\}$. A good way of thinking about L is that it is just an assignment of truth values to all the propositional atoms, as it was the case for propositional logic (we called that a valuation). The difference now is that we have more than one state, so this assignment depends on which state s the system is in: $L(s)$ contains all atoms which are true in state s.

定义: path

A *path* in a model $\mathcal{M} = (S, \rightarrow, L)$ is an infinite sequence of states $s_1, s_2, s_3, \ldots$ in S such that, for each $i \geq 1, s_i \rightarrow s_{i+1}$. We write the path as $s_1 \rightarrow s_2 \rightarrow \ldots$

定义: π^i

Consider the path $\pi = s_1 \rightarrow s_2 \rightarrow \ldots$.

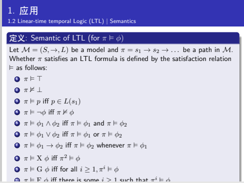
- It represents a possible future of our system: first it is in state s_1, then it is in state s_2, and so on.

We write π^i for the *suffix* starting at s_i, e.g., $s_3 \rightarrow s_4 \rightarrow \ldots$

定义: Semantic of LTL (for $\pi \vDash \phi$)

Let $\mathcal{M} = (S, \rightarrow, L)$ be a model and $\pi = s_1 \rightarrow s_2 \rightarrow \ldots$ be a path in $\mathcal{M}$. Whether π satisfies an LTL formula is defined by the satisfaction relation $\vDash$ as follows:

1. $\pi \vDash \top$
2. $\pi \nvDash \bot$
3. $\pi \vDash p$ iff $p \in L(s_1)$
4. $\pi \vDash \neg\phi$ iff $\pi \nvDash \phi$
5. $\pi \vDash \phi_1 \wedge \phi_2$ iff $\pi \vDash \phi_1$ and $\pi \vDash \phi_2$
6. $\pi \vDash \phi_1 \vee \phi_2$ iff $\pi \vDash \phi_1$ or $\pi \vDash \phi_2$
7. $\pi \vDash \phi_1 \rightarrow \phi_2$ iff $\pi \vDash \phi_2$ whenever $\pi \vDash \phi_1$
8. $\pi \vDash \mathrm{X}\ \phi$ iff $\pi^2 \vDash \phi$
9. $\pi \vDash \mathrm{G}\ \phi$ iff for all $i \geq 1, \pi^i \vDash \phi$

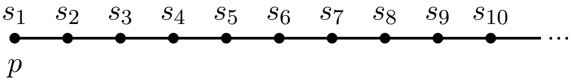


一种易于理解的方法

- X: next
- G: global
- F: future
- U: until (strong version)
- W: until (weak version)
- R: release

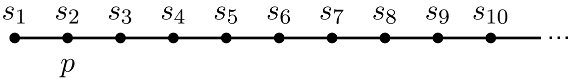
3. $\pi \vDash p$ iff $p \in L(s_1)$

For example, $\pi \vDash p$:



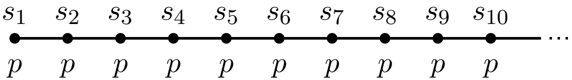
8. $\pi \vDash \mathrm{X}\ \phi$ iff $\pi^2 \vDash \phi$

For example, $\pi \vDash \mathrm{X}\ p$:



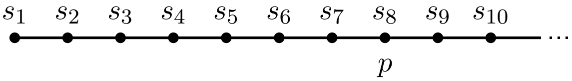
9. $\pi \vDash \mathrm{G}\ \phi$ iff for all $i \geq 1, \pi^i \vDash \phi$

For example, $\pi \vDash \mathrm{G}\ p$:



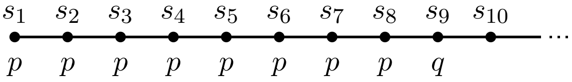
10. $\pi \vDash \mathrm{F}\ \phi$ iff there is some $i \geq 1$ such that $\pi^i \vDash \phi$

For example, $\pi \vDash \mathrm{F}\ p$:

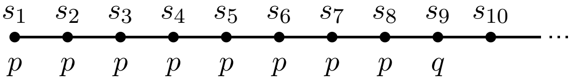


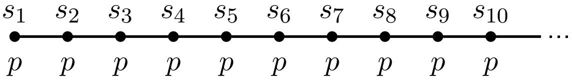
11. $\pi \vDash \phi \text{ U } \psi$ iff there is some $i \geq 1$ such that $\pi^i \vDash \psi$ and for all $j = 1, \ldots, i-1$ we have $\pi^j \vDash \phi$

For example, $\pi \vDash p \text{ U } q$:



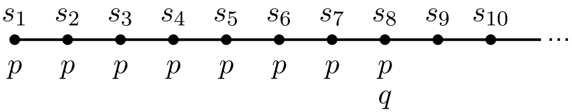
12. $\pi \vDash \phi \text{ W } \psi$ iff either there is some $i \geq 1$ such that $\pi^i \vDash \psi$ and for all $j = 1, \ldots, i-1$ we have $\pi^j \vDash \phi$; or for $k \geq 1$ we have $\pi^k \vDash \phi$
For example, $\pi \vDash p \text{ W } q$:

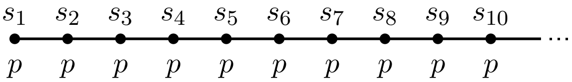




11. $\pi \vDash \phi \mathrm{~R~} \psi$ iff either there is some $i \geq 1$ such that $\pi^i \vDash \phi$ and for all $j = 1, \ldots, i$, we have $\pi^j \vDash \psi$, or for all $k \geq 1$ we have $\pi^k \vDash \psi$

For example, $\pi \vDash q \mathrm{~R~} p$:





性质: $\phi \mathsf{~R~} \psi \equiv \neg(\neg\phi \mathsf{~U~} \neg\psi)$

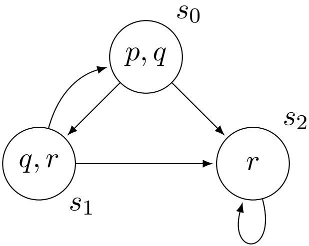
回顾定义: Semantic of LTL (for $\pi \vDash \phi$)

Let $\mathcal{M} = (S, \rightarrow, L)$ be a model and $\pi = s_1 \rightarrow s_2 \rightarrow \dots$ be a path in $\mathcal{M}$. Whether π satisfies an LTL formula is defined by the satisfaction relation $\vDash$ as follows:

...

定义: Semantic of LTL (for $\mathcal{M}, s \vDash \phi$)

Suppose $\mathcal{M} = (S, \rightarrow, L)$ is a model, $s \in S$, and ϕ an LTL formula. We write $\mathcal{M}, s \vDash \phi$ if, for every execution path π of $\mathcal{M}$ starting at s, we have $\pi \vDash \phi$.



- $\mathcal{M}, s_0 \vDash p \wedge q$ holds
- $\mathcal{M}, s_0 \vDash \neg r$ holds
- $\mathcal{M}, s_0 \vDash \top$ holds
- $\mathcal{M}, s_0 \vDash \mathrm{X}\ r$ holds
- $\mathcal{M}, s_0 \vDash \mathrm{X}\ (q \wedge r)$ does not hold
- $\mathcal{M}, s_0 \vDash \mathrm{G}\ \neg(p \wedge r)$ holds
- $\mathcal{M}, s_2 \vDash \mathrm{G}\ r$ holds
- For any state s of $\mathcal{M}$, we have $\mathcal{M}, s \vDash \mathrm{F}\ (\neg q \wedge r) \rightarrow \mathrm{F}\ \mathrm{G}\ r$

- Which π satisfies $\pi \vDash \mathrm{G}\ \mathrm{F}\ p$?
  - $\pi_1 = s_0 \rightarrow s_1 \rightarrow s_0 \rightarrow s_1 \rightarrow \ldots$ Yes
  - $\pi_2 = s_0 \rightarrow s_2 \rightarrow s_2 \rightarrow s_2 \rightarrow \ldots$ No
- $\mathcal{M}, s_0 \vDash \mathrm{G}\ \mathrm{F}\ p \rightarrow \mathrm{G}\ \mathrm{F}\ r$ holds
- $\mathcal{M}, s_0 \vDash \mathrm{G}\ \mathrm{F}\ r \rightarrow \mathrm{G}\ \mathrm{F}\ p$ does not hold

问题: 怎样将 LTL 用于常见的 Specification 的设计?
答: 看如下案例

- It is impossible to get to a state where *started* holds, but *ready* does not hold:

$$G\neg(\text{started} \wedge \neg\text{ready})$$

- For any state, if a *request* (of some resource) occurs, then it will eventually be *acknowledged*:

$$G\ (\text{requested} \rightarrow F\ \text{acknowledged})$$

- A certain process is *enabled* infinitely often on every computation path:

$$G\ F\ \text{enabled}$$

问题: 怎样将 LTL 用于常见的 Specification 的设计?
答: 看如下案例

- Whatever happens, a certain process will eventually be permanently *deadlocked*:

$$\mathrm{F}\ \mathrm{G}\ \mathrm{deadlock}$$

- If the process is enabled infinitely often, then it runs infinitely often:

$$\mathrm{G}\ \mathrm{F}\ \mathrm{enabled}\ \rightarrow \mathrm{G}\ \mathrm{F}\ \mathrm{running}$$

- An upwards travelling lift at the second floor does not change its direction when it has passengers wishing to go to the fifth floor:

$$\mathrm{G}\ (\mathrm{floor2} \wedge \mathrm{directionup} \wedge \mathrm{ButtonPressed5} \rightarrow (\mathrm{directionup}\ \mathrm{U}\ \mathrm{floor5}))$$

新的问题: 哪些 Specification**不能**用 LTL 来设计?
答: 看如下案例

- From any state it is possible to get to a restart state
  - i.e., there is a path from all states to a state satisfying restart
- The lift can remain idle on the third floor with its doors closed
  - i.e., from the state in which it is on the third floor, there is a path along which it stays there

LTL can't express these because it *cannot* directly assert the *existence* of paths.
怎么办?
另一种选择: CTL

$$\neg(\phi \wedge \psi) \equiv \neg\phi \vee \neg\psi \qquad \neg(\phi \vee \psi) \equiv \neg\phi \wedge \neg\psi$$

$$\neg \mathrm{G}\ \phi \equiv \mathrm{F}\ \neg\phi \qquad \neg \mathrm{F}\ \phi \equiv \mathrm{G}\ \neg\phi \qquad \neg \mathrm{X}\ \phi \equiv \mathrm{X}\ \neg\phi$$

$$\neg(\phi\ \mathrm{U}\ \psi) \equiv \neg\phi\ \mathrm{R}\ \neg\psi \qquad \neg(\phi\ \mathrm{R}\ \psi) \equiv \neg\phi\ \mathrm{U}\ \neg\psi$$

$$\mathrm{F}\ (\phi \vee \psi) \equiv \mathrm{F}\ \phi \vee \mathrm{F}\ \psi \qquad \mathrm{G}\ (\phi \wedge \psi) \equiv \mathrm{G}\ \phi \wedge \mathrm{G}\ \psi$$

$$\mathrm{F}\ \phi \equiv \top\ \mathrm{U}\ \phi \qquad \mathrm{G}\ \phi \equiv \bot\ \mathrm{R}\ \phi$$

$$\phi\ \mathrm{U}\ \psi \equiv \phi\ \mathrm{W}\ \psi \wedge \mathrm{F}\ \psi \qquad \phi\ \mathrm{W}\ \psi \equiv \phi\ \mathrm{U}\ \psi \vee \mathrm{G}\ \psi$$

$$\phi\ \mathrm{W}\ \psi \equiv \psi\ \mathrm{R}\ (\phi \vee \psi) \qquad \phi\ \mathrm{R}\ \psi \equiv \psi\ \mathrm{W}\ (\phi \wedge \psi)$$

回顾: 定义: Linear-time temporal logic (LTL)

Linear-time temporal logic (*LTL*) has following syntax given in BNF:

$$
\begin{aligned}
\phi ::= & \top \mid \bot \mid p \mid (\neg\phi) \mid (\phi \wedge \phi) \mid (\phi \vee \phi) \mid (\phi \rightarrow \phi) \\
& \mid (\mathrm{X}\ \phi) \mid (\mathrm{F}\ \phi) \mid (\mathrm{G}\ \phi) \mid (\phi\ \mathrm{U}\ \phi) \mid (\phi\ \mathrm{W}\ \phi) \mid (\phi\ \mathrm{R}\ \phi)
\end{aligned}
$$

where p is any propositional atom from some set **Atoms**.

定义: Computation Tree Logic (CTL)

Computation Tree logic (*CTL*) has following syntax given in BNF:

$$
\begin{aligned}
\phi ::= & \top \mid \bot \mid p \mid (\neg\phi) \mid (\phi \wedge \phi) \mid (\phi \vee \phi) \mid (\phi \rightarrow \phi) \\
& \mid (\mathrm{AX}\ \phi) \mid (\mathrm{EX}\ \phi) \mid (\mathrm{AF}\ \phi) \mid (\mathrm{EF}\ \phi) \mid (\mathrm{AG}\ \phi) \mid (\mathrm{EG}\ \phi) \\
& \mid \mathrm{A}[\phi\ \mathrm{U}\ \phi] \mid \mathrm{E}[\phi\ \mathrm{U}\ \phi]
\end{aligned}
$$

where p is any propositional atom from some set **Atoms**.

定义: Semantic of CTL (for $\mathcal{M}, s \models \phi$)

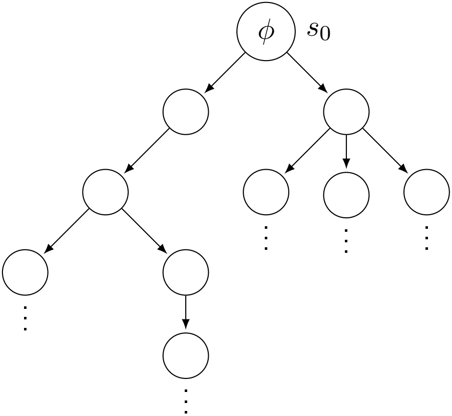
Let $\mathcal{M} = (S, \rightarrow, L)$ be a model for CTL, s in S, ϕ a CTL formula. The relation $\mathcal{M}, s \models \phi$ is defined by structural induction on ϕ

1. $\mathcal{M}, s \models \top$
2. $\mathcal{M}, s \not\models \bot$
3. $\mathcal{M}, s \models p$ iff $p \in L(s)$
4. $\mathcal{M}, s \models \neg\phi$ iff $\mathcal{M}, s \not\models \phi$
5. $\mathcal{M}, s \models \phi_1 \wedge \phi_2$ iff $\mathcal{M}, s \models \phi_1$ and $\mathcal{M}, s \models \phi_2$
6. $\mathcal{M}, s \models \phi_1 \vee \phi_2$ iff $\mathcal{M}, s \models \phi_1$ or $\mathcal{M}, s \models \phi_2$
7. $\mathcal{M}, s \models \phi_1 \rightarrow \phi_2$ iff $\mathcal{M}, s \models \phi_2$ whenever $\mathcal{M}, s \models \phi_1$
8. $\mathcal{M}, s \models \mathrm{AX}\ \phi$ iff *for all* s_1 such that $s \rightarrow s_1$ we have $\mathcal{M}, s_1 \models \phi$
9. $\mathcal{M}, s \models \mathrm{EX}\ \phi$ iff *for some* s_1 such that $s \rightarrow s_1$, we have $\mathcal{M}, s_1 \models \phi$
10. $\mathcal{M}, s \models \mathrm{AG}\ \phi$ iff *for all paths* $s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow \ldots$, where s_1 equals

3. $\mathcal{M}, s \vDash p$ iff $p \in L(s)$

For example,

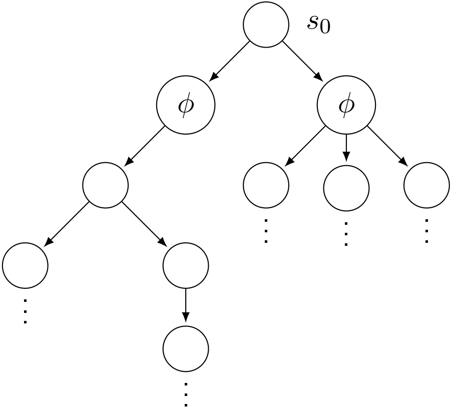
$$\mathcal{M}, s_0 \vDash \phi$$



$\mathcal{M}, s \vDash \mathrm{AX}\ \phi$ iff *for all* s_1 such that $s \to s_1$ we have $\mathcal{M}, s_1 \vDash \phi$

For example,

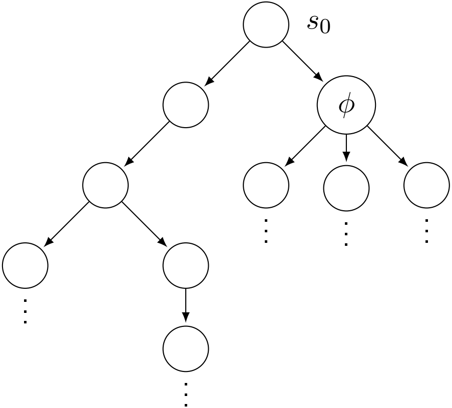
$$\mathcal{M}, s_0 \vDash \mathrm{AX}\ \phi$$



$\mathcal{M}, s \models \mathrm{EX}\ \phi$ iff *for some* s_1 such that $s \to s_1$, we have $\mathcal{M}, s_1 \models \phi$

For example,

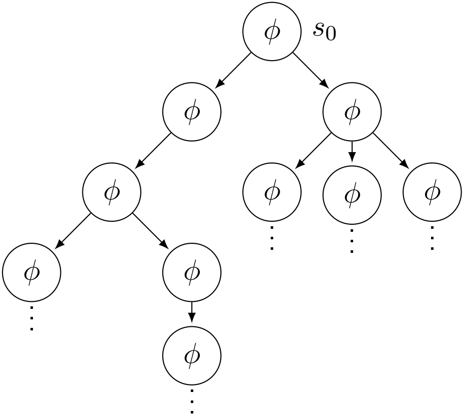
$$\mathcal{M}, s_0 \models \mathrm{EX}\ \phi$$



$\mathcal{M}, s \vDash \mathrm{AG}\ \phi$ iff *for all paths* $s_1 \to s_2 \to s_3 \to \ldots$, where s_1 equals s, and *for all* s_i along the path, we have $\mathcal{M}, s_i \vDash \phi$

For example,

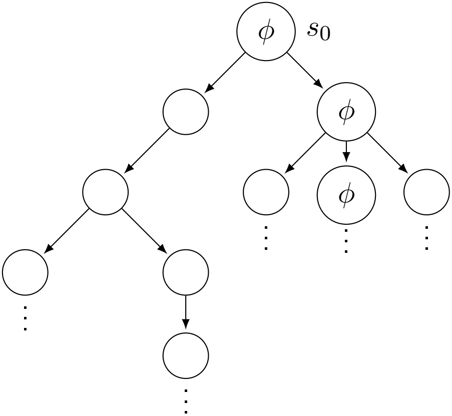
$$\mathcal{M}, s_0 \vDash \mathrm{AG}\ \phi$$



$\mathcal{M}, s \vDash \mathrm{EG}\ \phi$ iff *there is a path* $s_1 \to s_2 \to s_3 \to \ldots$, where s_1 equals s, and *for all* s_i along the path, we have $\mathcal{M}, s_i \vDash \phi$

For example,

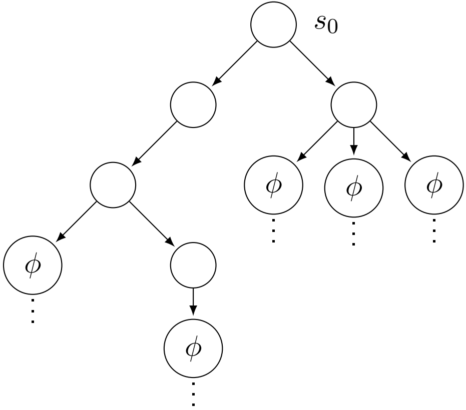
$$\mathcal{M}, s_0 \vDash \mathrm{EG}\ \phi$$



$\mathcal{M}, s \vDash \mathrm{AF}\ \phi$ iff *for all* paths $s_1 \to s_2 \to s_3 \to \ldots$, where s_1 equals s, and *there is* some s_i such that $\mathcal{M}, s_i \vDash \phi$

For example,

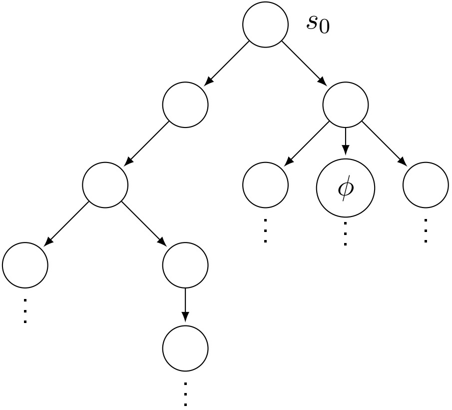
$$\mathcal{M}, s_0 \vDash \mathrm{AF}\ \phi$$



$\mathcal{M}, s \vDash \mathrm{EF}\ \phi$ iff *there is a path* $s_1 \to s_2 \to s_3 \to \ldots,$ where s_1 equals s, and *there is* some s_i such that $\mathcal{M}, s_i \vDash \phi$

For example,

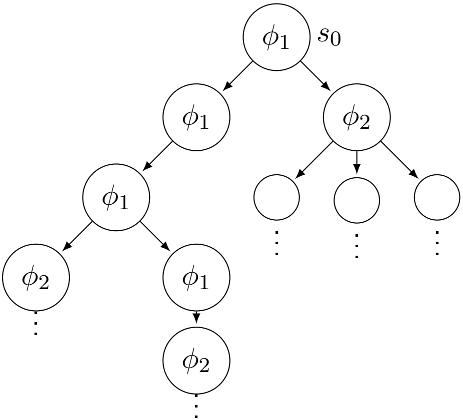
$$\mathcal{M}, s_0 \vDash \mathrm{EF}\ \phi$$



$\mathcal{M}, s \vDash A[\phi_1 \ \mathrm{U}\ \phi_2]$ iff *for all paths* $s_1 \to s_2 \to s_3 \to \ldots,$ where s_1 equals s, that path satisfies $\phi_1 \ \mathrm{U}\ \phi_2$, i.e., there is some s_i along the path, such that $\mathcal{M}, s_i \vDash \phi_2$, and, for each $j < i$, we have $\mathcal{M}, s_i \vDash \phi_1$

For example,

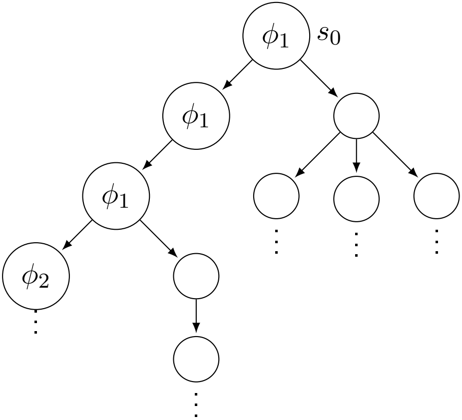
$$\mathcal{M}, s_0 \vDash A[\phi_1 \ \mathrm{U}\ \phi_2]$$

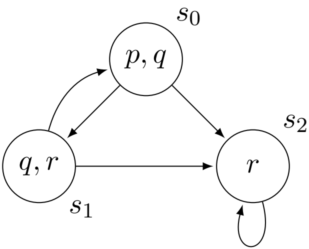


$\mathcal{M}, s \vDash E[\phi_1 \ \mathrm{U} \ \phi_2]$ iff *there is a path* $s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow \ldots$, where s_1 equals s, that path satisfies $\phi_1 \ \mathrm{U} \ \phi_2$, i.e., there is some s_i along the path, such that $\mathcal{M}, s_i \vDash \phi_2$, and, for each $j < i$, we have $\mathcal{M}, s_i \vDash \phi_1$

For example,

$$\mathcal{M}, s_0 \vDash E[\phi_1 \ \mathrm{U} \ \phi_2]$$

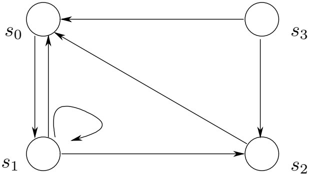




- $\mathcal{M}, s_0 \models p \wedge q$ holds
- $\mathcal{M}, s_0 \models \neg r$ holds
- $\mathcal{M}, s_0 \models \top$ holds
- $\mathcal{M}, s_0 \models \mathrm{EX}\ (q \wedge r)$ holds
- $\mathcal{M}, s_0 \models \neg\mathrm{AX}\ (q \wedge r)$ holds
- $\mathcal{M}, s_0 \models \neg\mathrm{EF}\ (p \wedge r)$ holds
- $\mathcal{M}, s_2 \models \mathrm{EG}\ r$ holds
- $\mathcal{M}, s_0 \models \mathrm{AF}\ r$ holds
- $\mathcal{M}, s_0 \models \mathrm{E}[(p \wedge q)\ \mathrm{U}\ r]$ holds
- $\mathcal{M}, s_0 \models \mathrm{A}[p\ \mathrm{U}\ r]$ holds
- $\mathcal{M}, s_0 \models \mathrm{AG}\ (p \vee q \vee r \rightarrow \mathrm{EF}\ \mathrm{EG}\ r)$ holds

回顾: 反例:

Given a set of states $A = \{s_0, s_1, s_2, s_3\}$, let $R^{\mathcal{M}}$ be the set $\{(s_0, s_1), (s_1, s_0), (s_1, s_1), (s_1, s_2), (s_2, s_0), (s_3, s_0), (s_3, s_2)\}$. We may depict this model as *a directed graph* in a figure, where an edge (a transition) leads from a node s to a node s' iff $(s, s') \in R^{\mathcal{M}}$.



回顾: 反例: How to define Reachability as ϕ

Given nodes n and n' in a directed graph, is there a finite path of transitions from n to n'?

回顾：反例：How to define Reachability as ϕ

Given nodes n and n' in a directed graph, is there a finite path of transitions from n to n'?

回顾：反例：一种答案

$$(u = v) \vee \exists x(R(u, x) \wedge R(x, v)) \vee \exists x_1 \exists x_2(R(u, x_1) \wedge R(x_1, x_2) \wedge R(x_2, v)) \vee \ldots$$

- This is infinite, so it's *not* a well-formed formula.
- Can we find a well-formed formula with the same meaning? *No!*

回顾：反例：How to define Reachability as ϕ

Given nodes n and n' in a directed graph, is there a finite path of transitions from n to n'?

回顾：另一种答案：Second-order Logic

$$\neg\exists P \forall x \forall y \forall z\ (C_1 \wedge C_2 \wedge C_3 \wedge C_4)$$

where

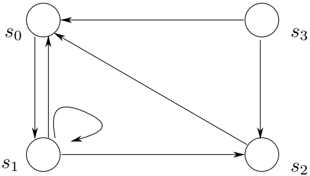
$$C_1 \stackrel{\text{def}}{=} P(x,x)$$
$$C_2 \stackrel{\text{def}}{=} P(x,y) \wedge P(y,z) \rightarrow P(x,z)$$
$$C_3 \stackrel{\text{def}}{=} P(u,v) \rightarrow \perp$$
$$C_4 \stackrel{\text{def}}{=} R(x,y) \rightarrow P(x,y)$$

回顾：反例： How to define Reachability as ϕ

Given nodes n and n' in a directed graph, is there a finite path of transitions from n to n'?

新答案：使用 CTL

$$\mathcal{M}, n \vDash \text{EF } (s = n')$$



$$\neg \mathrm{AF}\ \phi \equiv \mathrm{EG}\ \neg\phi$$

$$\neg \mathrm{EF}\ \phi \equiv \mathrm{AG}\ \neg\phi$$

$$\neg \mathrm{AX}\ \phi \equiv \mathrm{EX}\ \neg\phi$$

$$\mathrm{AF}\ \phi \equiv \mathrm{A}[\top\ \mathrm{U}\ \phi]$$

$$\mathrm{EF}\ \phi \equiv \mathrm{E}[\top\ \mathrm{U}\ \phi]$$

回顾: LTL cannot express these because it *cannot* directly assert the *existence* of paths.

- CTL can express the *existence* of paths.

新的问题: Is CTL better than LTL? i.e., Is LTL a subset of CTL?
答案: *No*

例:
An LTL formula:

$$\mathrm{FG}\ p$$

How to express it in CTL? $\mathrm{AFAG}\ p$? *No*
Another LTL formula?

$$\mathrm{F}\ p \to \mathrm{F}\ q$$

怎么办?

*CTL** is a logic which *combines* the expressive powers of *LTL* and *CTL*, by *dropping* the CTL *constraint* that every temporal operator (X, U, F, G) has to be associated with a unique path quantifier (A, E). For example:

- A[(p U r) $\vee$ (q U r)]: along all paths, either p is true until r, or q is true until r.
- A[X p $\vee$ XX p]: along all paths, p is true in the next state, or the next but one.
- E[G F p]: there is a path along which p is infinitely often true.

定义: CTL*

The syntax of CTL* involves two classes of formulas:

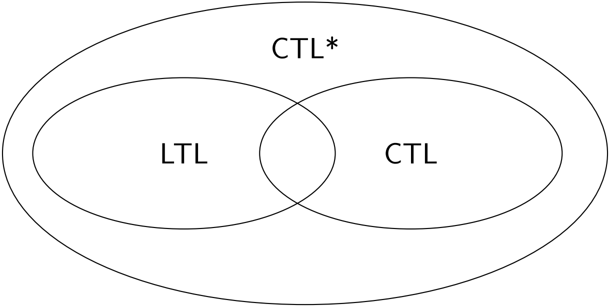
- *state formulas*, which are evaluated in states:

$$\phi ::== \top \mid p \mid (\neg\phi) \mid (\phi \wedge \phi) \mid \mathrm{A}[\alpha] \mid \mathrm{E}[\alpha]$$

  where p is any atomic formula and α any path formula

- *path formulas*, which are evaluated along paths:

$$\alpha ::== \phi \mid (\neg\alpha) \mid (\alpha \wedge \alpha) \mid (\alpha \ \mathrm{U} \ \alpha) \mid (\mathrm{G} \ \alpha) \mid (\mathrm{F} \ \alpha) \mid (\mathrm{X} \ \alpha)$$



CTL*
LTL
CTL

Concluding:

- LTL, CTL, CTL* can be used to model ϕ, instead of propositional logics, first-order logics, higher-order logics
- Transition system can be used to model $\mathcal{M}$, instead of logics

剩下的问题:

- How to program using LTL, CTL, CTL*, transition system
  - Using NuSMV (见第 1.4 节)
- How to implement algorithms for NuSMV
  - (见第 2 节)

作业

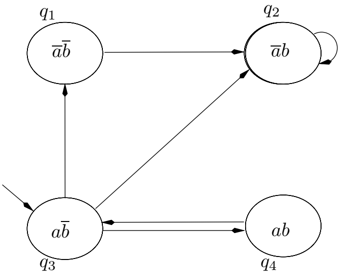


Figure 3.39. A model $\mathcal{M}$.

2. Consider the system of Figure 3.39. For each of the formulas ϕ:
   (a) $\mathrm{G}\, a$
   (b) $a \mathrel{\mathrm{U}} b$
   (c) $a \mathrel{\mathrm{U}} \mathrm{X}\,(a \wedge \neg b)$
   (d) $\mathrm{X}\,\neg b \wedge \mathrm{G}\,(\neg a \vee \neg b)$
   (e) $\mathrm{X}\,(a \wedge b) \wedge \mathrm{F}\,(\neg a \wedge \neg b)$
      (i) Find a path from the initial state q_3 which satisfies ϕ.
      (ii) Determine whether $\mathcal{M}, q_3 \vDash \phi$.

作业

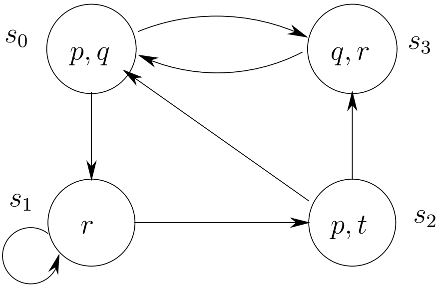


Figure 3.41. Another model with four states.

8. Consider the model $\mathcal{M}$ in Figure 3.41. Check whether $\mathcal{M}, s_0 \vDash \phi$ and $\mathcal{M}, s_2 \vDash \phi$ hold for the CTL formulas ϕ:
   (a) $\mathrm{AF}\, q$
   (b) $\mathrm{AG}\,(\mathrm{EF}\,(p \vee r))$
   (c) $\mathrm{EX}\,(\mathrm{EX}\, r)$
   (d) $\mathrm{AG}\,(\mathrm{AF}\, q)$.

本章大作业参考论文

大作业可参考论文 (但不限于下列论文):

- 经典
  - Word level model checking—avoiding the Pentium FDIV error
- 应用
  - Liveness Verification of Stateful Network Functions
  - Weak, strong, and strong cyclic planning via symbolic model checking
  - Specification Patterns for Robotic Missions
  - Synthesis of Reactive Switching Protocols From Temporal Logic Specifications
- 工具实现
  - NUSMV:A new symbolic model verifier
  - The nuXmv symbolic model checker