

形式化方法导引

第 5 章 模型检测

5.1 应用

黄文超

<http://staff.ustc.edu.cn/~huangwc/fm.html>

1. 应用

1.1. Model Checking | 回顾

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

1. 应用

1.1. Model Checking | 回顾

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

1. 应用

1.1. Model Checking | 回顾

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

1. 应用

1.1. Model Checking | 回顾

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*:

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

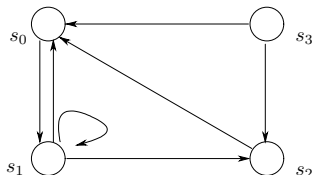
- \mathcal{M} is some sort of situation or *model* of a system
- ϕ is a *specification*, a formula of that logic, expressing what should be true in situation \mathcal{M} .
- At the *heart* of this set-up is that one can often *specify and implement algorithms* for computing \models .

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

1. 应用

1.1. Model Checking | 回顾 | Limitation of first-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

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

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

1. 应用

1.1. Model Checking | 回顾 | Limitation of higher-order logic

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

$$C_1 \stackrel{\text{def}}{=} P(x, x)$$

$$C_2 \stackrel{\text{def}}{=} P(x, y) \wedge P(y, z) \rightarrow P(x, z)$$

$$C_3 \stackrel{\text{def}}{=} P(u, v) \rightarrow \perp$$

$$C_4 \stackrel{\text{def}}{=} R(x, y) \rightarrow P(x, y)$$

问题:

- 难以理解 ϕ , 难以构建 ϕ
- 如何自动验证?

1. 应用

1.1. Model Checking | 回顾 | Limitation of higher-order logic

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

$$C_1 \stackrel{\text{def}}{=} P(x, x)$$

$$C_2 \stackrel{\text{def}}{=} P(x, y) \wedge P(y, z) \rightarrow P(x, z)$$

$$C_3 \stackrel{\text{def}}{=} P(u, v) \rightarrow \perp$$

$$C_4 \stackrel{\text{def}}{=} R(x, y) \rightarrow P(x, y)$$

问题:

- 难以理解 ϕ , 难以构建 ϕ
- 如何自动验证?

1. 应用

1.1. Model Checking | 回顾 | Limitation of higher-order logic

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

$$C_1 \stackrel{\text{def}}{=} P(x, x)$$

$$C_2 \stackrel{\text{def}}{=} P(x, y) \wedge P(y, z) \rightarrow P(x, z)$$

$$C_3 \stackrel{\text{def}}{=} P(u, v) \rightarrow \perp$$

$$C_4 \stackrel{\text{def}}{=} R(x, y) \rightarrow P(x, y)$$

问题:

- 难以理解 ϕ , 难以构建 ϕ
- 如何自动验证?

1. 应用

1.1. Model Checking | 回顾 | Limitation of higher-order logic

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

$$C_1 \stackrel{\text{def}}{=} P(x, x)$$

$$C_2 \stackrel{\text{def}}{=} P(x, y) \wedge P(y, z) \rightarrow P(x, z)$$

$$C_3 \stackrel{\text{def}}{=} P(u, v) \rightarrow \perp$$

$$C_4 \stackrel{\text{def}}{=} R(x, y) \rightarrow P(x, y)$$

问题:

- 难以理解 ϕ , 难以构建 ϕ
- 如何自动验证?

1. 应用

1.1. Model Checking | 回顾 | Limitation of Logics

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?

1. 应用

1.1. Model Checking | 回顾 | Limitation of Logics

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?

1. 应用

1.1. Model Checking | 回顾 | Limitation of Logics

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?

1. 应用

1.1. Model Checking | 回顾 | Limitation of Logics

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?

1. 应用

1.1. Model Checking | 回顾 | Limitation of Logics

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?

1. 应用

1.1. Model Checking | 本章内容

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 | 本章内容

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 | 本章内容

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 | 本章内容

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 | 本章内容

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 | 本章内容

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 | 本章内容

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 | 本章内容

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

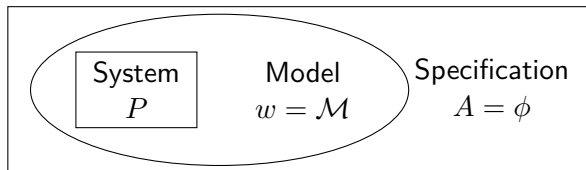
再往前回顾: 定义: Verifier

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

1. 应用

1.1. Model Checking | 定义

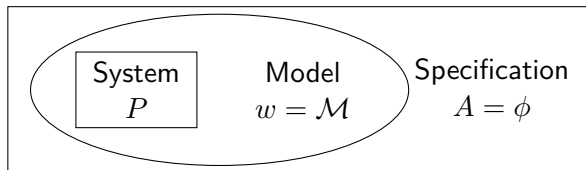
再往前回顾: 定义: Verifier

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

1. 应用

1.1. Model Checking | 定义

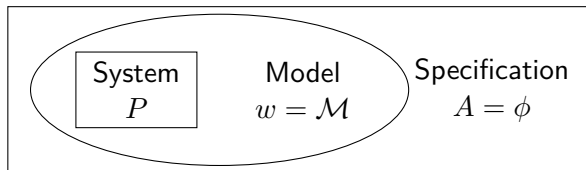
再往前回顾: 定义: Verifier

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

1. 应用

1.1. Model Checking | 定义

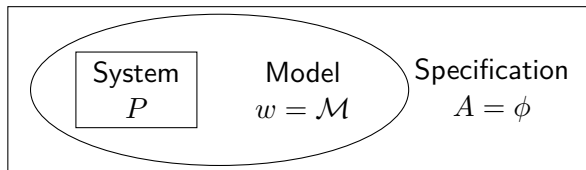
再往前回顾: 定义: Verifier

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

1. 应用

1.1. Model Checking | 定义

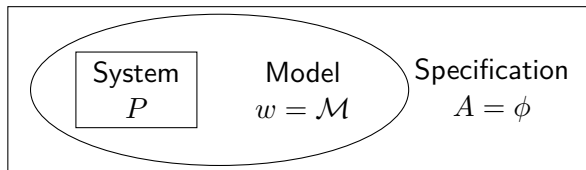
再往前回顾: 定义: Verifier

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

1. 应用

1.1. Model Checking | 定义

定义: Model checking

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

- \mathcal{M} is an *appropriate* model of the system under consideration.
- s is a *state* of that model
- \models 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?

1. 应用

1.1. Model Checking | 定义

定义: Model checking

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

- \mathcal{M} is an *appropriate* model of the system under consideration.
- s is a *state* of that model
- \models 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?

1. 应用

1.1. Model Checking | 定义

定义: Model checking

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

- \mathcal{M} is an *appropriate* model of the system under consideration.
- s is a *state* of that model
- \models 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?

1. 应用

1.1. Model Checking | 定义

定义: Model checking

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

- \mathcal{M} is an *appropriate* model of the system under consideration.
- s is a *state* of that model
- \models 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?

1. 应用

1.1. Model Checking | 定义

定义: Model checking

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

- \mathcal{M} is an *appropriate* model of the system under consideration.
- s is a *state* of that model
- \models 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?

1. 应用

1.1. Model Checking | 定义

定义: Model checking

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

- \mathcal{M} is an *appropriate* model of the system under consideration.
- s is a *state* of that model
- \models 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?

1. 应用

1.1. Model Checking | 定义

定义: Model checking

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

- \mathcal{M} is an *appropriate* model of the system under consideration.
- s is a *state* of that model
- \models 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?

1. 应用

1.1. Model Checking | 定义

定义: Model checking

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

- \mathcal{M} is an *appropriate* model of the system under consideration.
- s is a *state* of that model
- \models 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?

1. 应用

1.1. Model Checking | 定义

定义: Model checking

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

- \mathcal{M} is an *appropriate* model of the system under consideration.
- s is a *state* of that model
- \models 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?

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:

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

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

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

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

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:

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

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

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

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

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:

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

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

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

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

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:

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

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

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

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

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:

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

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

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

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

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:

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

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

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

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

1. 应用

1.2 Linear-time temporal Logic (LTL) | Semantics

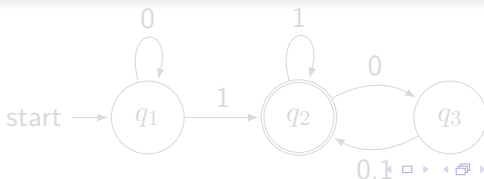
问题: 定义 (理解) 新的符号? 如: X, F, G, U, W, R

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 $\delta : Q \times \Sigma \rightarrow Q$ is the *transition function*,
- 4 $q_0 \in Q$ is the *start state*, and
- 5 $F \subseteq Q$ is the set of *accept states*.



1. 应用

1.2 Linear-time temporal Logic (LTL) | Semantics

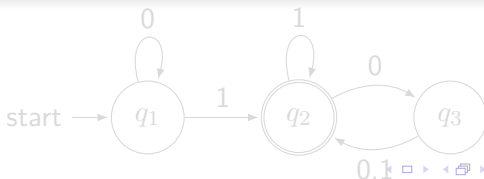
问题: 定义 (理解) 新的符号? 如: X, F, G, U, W, R

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 $\delta : Q \times \Sigma \rightarrow Q$ is the *transition function*,
- 4 $q_0 \in Q$ is the *start state*, and
- 5 $F \subseteq Q$ is the set of *accept states*.



1. 应用

1.2 Linear-time temporal Logic (LTL) | Semantics

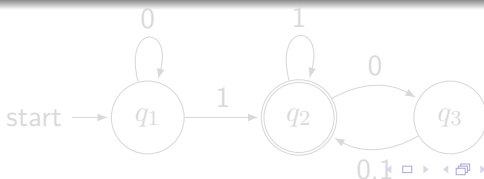
问题: 定义 (理解) 新的符号? 如: X, F, G, U, W, R

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 $\delta : Q \times \Sigma \rightarrow Q$ is the *transition function*,
- 4 $q_0 \in Q$ is the *start state*, and
- 5 $F \subseteq Q$ is the set of *accept states*.



1. 应用

1.2 Linear-time temporal Logic (LTL) | Semantics

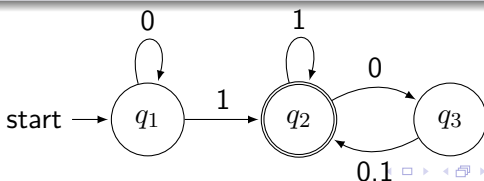
问题: 定义 (理解) 新的符号? 如: X, F, G, U, W, R

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 $\delta : Q \times \Sigma \rightarrow Q$ is the *transition function*,
- 4 $q_0 \in Q$ is the *start state*, and
- 5 $F \subseteq Q$ is the set of *accept states*.



1. 应用

1.2 Linear-time temporal Logic (LTL) | Semantics

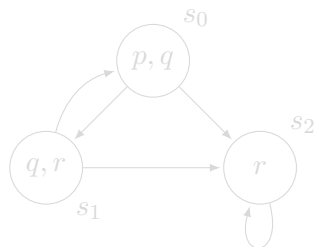
问题: 定义 (理解) 新的符号? 如: X, F, G, U, W, R

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 $\mathcal{M} = (S, \rightarrow, L)$ is

- S : a set of states
 - \rightarrow : a transition relation.
 - every $s \in S$ has some $s' \in S$ with $s \rightarrow s'$
 - L : a label function.
 - $L : S \rightarrow \mathcal{P}(\text{Atoms})$
-
- $S = \{s_0, s_1, s_2\}$
 - transitions: $s_0 \rightarrow s_1, s_0 \rightarrow s_2, s_1 \rightarrow s_0, s_1 \rightarrow s_2, s_2 \rightarrow s_2$
 - $L(s_0) = \{p, q\}, L(s_1) = \{q, r\}, L(s_2) = \{r\}$



1. 应用

1.2 Linear-time temporal Logic (LTL) | Semantics

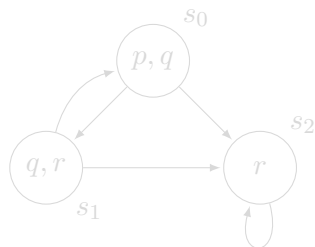
问题: 定义 (理解) 新的符号? 如: X, F, G, U, W, R

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 $\mathcal{M} = (S, \rightarrow, L)$ is

- S : a set of states
 - \rightarrow : a transition relation.
 - every $s \in S$ has some $s' \in S$ with $s \rightarrow s'$
 - L : a label function.
 - $L : S \rightarrow \mathcal{P}(\text{Atoms})$
-
- $S = \{s_0, s_1, s_2\}$
 - transitions: $s_0 \rightarrow s_1, s_0 \rightarrow s_2, s_1 \rightarrow s_0, s_1 \rightarrow s_2, s_2 \rightarrow s_2$
 - $L(s_0) = \{p, q\}, L(s_1) = \{q, r\}, L(s_2) = \{r\}$



1. 应用

1.2 Linear-time temporal Logic (LTL) | Semantics

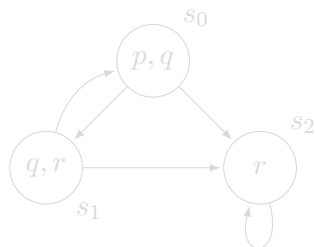
问题: 定义 (理解) 新的符号? 如: X, F, G, U, W, R

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 $\mathcal{M} = (S, \rightarrow, L)$ is

- S : a set of states
 - \rightarrow : a transition relation.
 - every $s \in S$ has some $s' \in S$ with $s \rightarrow s'$
 - L : a label function.
 - $L : S \rightarrow \mathcal{P}(\text{Atoms})$
-
- $S = \{s_0, s_1, s_2\}$
 - transitions: $s_0 \rightarrow s_1, s_0 \rightarrow s_2, s_1 \rightarrow s_0, s_1 \rightarrow s_2, s_2 \rightarrow s_2$
 - $L(s_0) = \{p, q\}, L(s_1) = \{q, r\}, L(s_2) = \{r\}$



1. 应用

1.2 Linear-time temporal Logic (LTL) | Semantics

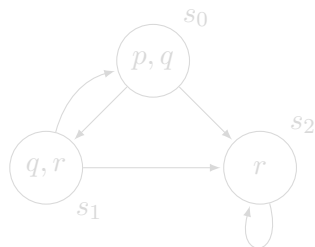
问题: 定义 (理解) 新的符号? 如: X, F, G, U, W, R

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 $\mathcal{M} = (S, \rightarrow, L)$ is

- S : a set of states
 - \rightarrow : a transition relation.
 - every $s \in S$ has some $s' \in S$ with $s \rightarrow s'$
 - L : a label function.
 - $L : S \rightarrow \mathcal{P}(\text{Atoms})$
-
- $S = \{s_0, s_1, s_2\}$
 - transitions: $s_0 \rightarrow s_1, s_0 \rightarrow s_2, s_1 \rightarrow s_0, s_1 \rightarrow s_2, s_2 \rightarrow s_2$
 - $L(s_0) = \{p, q\}, L(s_1) = \{q, r\}, L(s_2) = \{r\}$



1. 应用

1.2 Linear-time temporal Logic (LTL) | Semantics

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

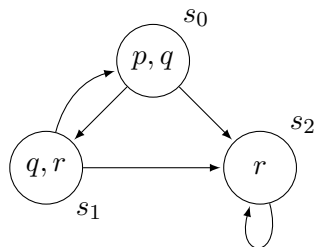
A transition system $\mathcal{M} = (S, \rightarrow, L)$ is

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

- $S = \{s_0, s_1, s_2\}$

- transitions: $s_0 \rightarrow s_1, s_0 \rightarrow s_2, s_1 \rightarrow s_0, s_1 \rightarrow s_2, s_2 \rightarrow s_2$

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



1. 应用

1.2 Linear-time temporal Logic (LTL) | Semantics

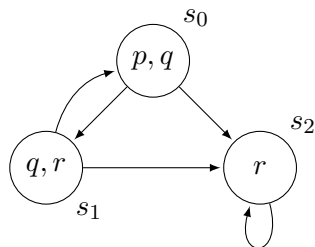
问题: 定义 (理解) 新的符号? 如: X, F, G, U, W, R

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 $\mathcal{M} = (S, \rightarrow, L)$ is

- S : a set of states
- \rightarrow : a transition relation.
 - every $s \in S$ has some $s' \in S$ with $s \rightarrow s'$
- L : a label function.
 - $L : S \rightarrow \mathcal{P}(\text{Atoms})$
- $S = \{s_0, s_1, s_2\}$
- transitions: $s_0 \rightarrow s_1, s_0 \rightarrow s_2, s_1 \rightarrow s_0, s_1 \rightarrow s_2, s_2 \rightarrow s_2$
- $L(s_0) = \{p, q\}, L(s_1) = \{q, r\}, L(s_2) = \{r\}$



1. 应用

1.2 Linear-time temporal Logic (LTL) | Semantics

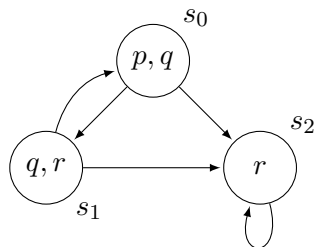
问题: 定义 (理解) 新的符号? 如: X, F, G, U, W, R

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 $\mathcal{M} = (S, \rightarrow, L)$ is

- S : a set of states
 - \rightarrow : a transition relation.
 - every $s \in S$ has some $s' \in S$ with $s \rightarrow s'$
 - L : a label function.
 - $L : S \rightarrow \mathcal{P}(\text{Atoms})$
-
- $S = \{s_0, s_1, s_2\}$
 - transitions: $s_0 \rightarrow s_1, s_0 \rightarrow s_2, s_1 \rightarrow s_0, s_1 \rightarrow s_2, s_2 \rightarrow s_2$
 - $L(s_0) = \{p, q\}, L(s_1) = \{q, r\}, L(s_2) = \{r\}$



1. 应用

1.2 Linear-time temporal Logic (LTL) | Semantics

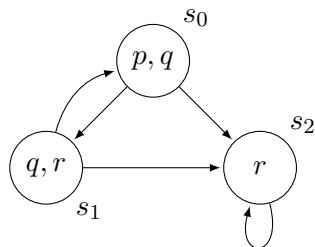
问题: 定义 (理解) 新的符号? 如: X, F, G, U, W, R

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 $\mathcal{M} = (S, \rightarrow, L)$ is

- S : a set of states
 - \rightarrow : a transition relation.
 - every $s \in S$ has some $s' \in S$ with $s \rightarrow s'$
 - L : a label function.
 - $L : S \rightarrow \mathcal{P}(\text{Atoms})$
-
- $S = \{s_0, s_1, s_2\}$
 - transitions: $s_0 \rightarrow s_1, s_0 \rightarrow s_2, s_1 \rightarrow s_0, s_1 \rightarrow s_2, s_2 \rightarrow s_2$
 - $L(s_0) = \{p, q\}, L(s_1) = \{q, r\}, L(s_2) = \{r\}$



1. 应用

1.2 Linear-time temporal Logic (LTL) | Semantics

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, \dots in S such that, for each $i \geq 1, s_i \rightarrow s_{i+1}$. We write the path as $s_1 \rightarrow s_2 \rightarrow \dots$

定义: π^i

Consider the path $\pi = s_1 \rightarrow s_2 \rightarrow \dots$

- 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., $s_3 \rightarrow s_4 \rightarrow \dots$

1. 应用

1.2 Linear-time temporal Logic (LTL) | Semantics

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, \dots in S such that, for each $i \geq 1, s_i \rightarrow s_{i+1}$. We write the path as $s_1 \rightarrow s_2 \rightarrow \dots$

定义: π^i

Consider the path $\pi = s_1 \rightarrow s_2 \rightarrow \dots$

- 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., $s_3 \rightarrow s_4 \rightarrow \dots$

1. 应用

1.2 Linear-time temporal Logic (LTL) | Semantics

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

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 \models as follows:

- 1 $\pi \models \top$
- 2 $\pi \not\models \perp$
- 3 $\pi \models p$ iff $p \in L(s_1)$
- 4 $\pi \models \neg\phi$ iff $\pi \not\models \phi$
- 5 $\pi \models \phi_1 \wedge \phi_2$ iff $\pi \models \phi_1$ and $\pi \models \phi_2$
- 6 $\pi \models \phi_1 \vee \phi_2$ iff $\pi \models \phi_1$ or $\pi \models \phi_2$
- 7 $\pi \models \phi_1 \rightarrow \phi_2$ iff $\pi \models \phi_2$ whenever $\pi \models \phi_1$

1. 应用

1.2 Linear-time temporal Logic (LTL) | Semantics

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

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 \models as follows:

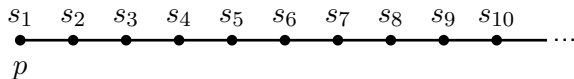
- ⑧ $\pi \models X \phi$ iff $\pi^2 \models \phi$
- ⑨ $\pi \models G \phi$ iff for all $i \geq 1, \pi^i \models \phi$
- ⑩ $\pi \models F \phi$ iff there is some $i \geq 1$ such that $\pi^i \models \phi$
- ⑪ $\pi \models \phi U \psi$ iff there is some $i \geq 1$ such that $\pi^i \models \psi$ and for all $j = 1, \dots, i - 1$ we have $\pi^j \models \phi$
- ⑫ $\pi \models \phi W \psi$ iff either there is some $i \geq 1$ such that $\pi^i \models \psi$ and for all $j = 1, \dots, i - 1$ we have $\pi^j \models \phi$; or for $k \geq 1$ we have $\pi^k \models \phi$
- ⑬ $\pi \models \phi R \psi$ iff either there is some $i \geq 1$ such that $\pi^i \models \phi$ and for all $j = 1, \dots, i$, we have $\pi^j \models \psi$, or for all $k \geq 1$ we have $\pi^k \models \psi$

1. 应用

1.2 Linear-time temporal Logic (LTL) | Semantics

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

For example, $\pi \models p$:



8. $\pi \models X \phi$ iff $\pi^2 \models \phi$

For example, $\pi \models X p$:

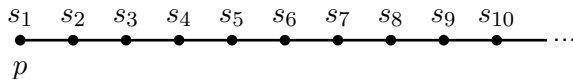


1. 应用

1.2 Linear-time temporal Logic (LTL) | Semantics

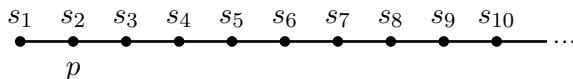
3. $\pi \models p$ iff $p \in L(s_1)$

For example, $\pi \models p$:



8. $\pi \models X \phi$ iff $\pi^2 \models \phi$

For example, $\pi \models X p$:

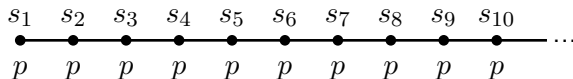


1. 应用

1.2 Linear-time temporal Logic (LTL) | Semantics

9. $\pi \models G \phi$ iff for all $i \geq 1, \pi^i \models \phi$

For example, $\pi \models G p$:



10. $\pi \models F \phi$ iff there is some $i \geq 1$ such that $\pi^i \models \phi$

For example, $\pi \models F p$:

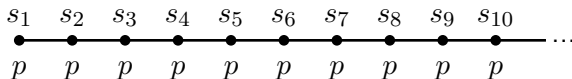


1. 应用

1.2 Linear-time temporal Logic (LTL) | Semantics

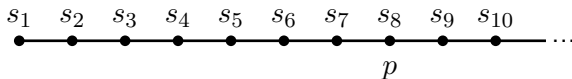
9. $\pi \models G \phi$ iff for all $i \geq 1, \pi^i \models \phi$

For example, $\pi \models G p$:



10. $\pi \models F \phi$ iff there is some $i \geq 1$ such that $\pi^i \models \phi$

For example, $\pi \models F p$:

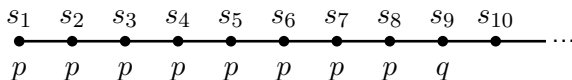


1. 应用

1.2 Linear-time temporal Logic (LTL) | Semantics

11. $\pi \models \phi \cup \psi$ iff there is some $i \geq 1$ such that $\pi^i \models \psi$ and for all $j = 1, \dots, i - 1$ we have $\pi^j \models \phi$

For example, $\pi \models p \cup q$:



12. $\pi \models \phi \text{ W } \psi$ iff either there is some $i \geq 1$ such that $\pi^i \models \psi$ and for all $j = 1, \dots, i - 1$ we have $\pi^j \models \phi$; or for $k \geq 1$ we have $\pi^k \models \psi$

For example, $\pi \models p \text{ W } q$:

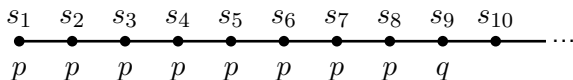


1. 应用

1.2 Linear-time temporal Logic (LTL) | Semantics

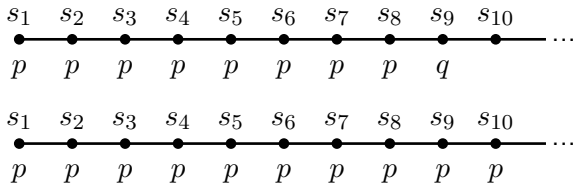
11. $\pi \models \phi \cup \psi$ iff there is some $i \geq 1$ such that $\pi^i \models \psi$ and for all $j = 1, \dots, i - 1$ we have $\pi^j \models \phi$

For example, $\pi \models p \cup q$:



12. $\pi \models \phi \text{ W } \psi$ iff either there is some $i \geq 1$ such that $\pi^i \models \psi$ and for all $j = 1, \dots, i - 1$ we have $\pi^j \models \phi$; or for $k \geq 1$ we have $\pi^k \models \phi$

For example, $\pi \models p \text{ W } q$:

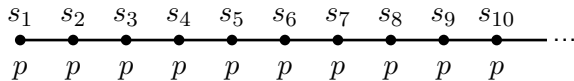
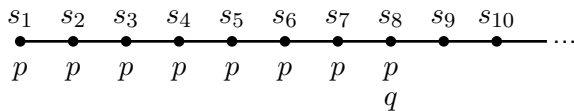


1. 应用

1.2 Linear-time temporal Logic (LTL) | Semantics

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

For example, $\pi \models q \text{ R } p$:



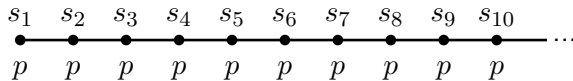
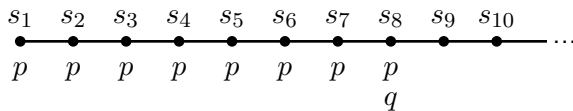
性质: $\phi \text{ R } \psi \equiv \neg(\neg\phi \text{ U } \neg\psi)$

1. 应用

1.2 Linear-time temporal Logic (LTL) | Semantics

11. $\pi \models \phi R \psi$ iff either there is some $i \geq 1$ such that $\pi^i \models \phi$ and for all $j = 1, \dots, i$, we have $\pi^j \models \psi$, or for all $k \geq 1$ we have $\pi^k \models \psi$

For example, $\pi \models q R p$:



性质: $\phi R \psi \equiv \neg(\neg\phi U \neg\psi)$

1. 应用

1.2 Linear-time temporal Logic (LTL) | Semantics

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

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 \models as follows:

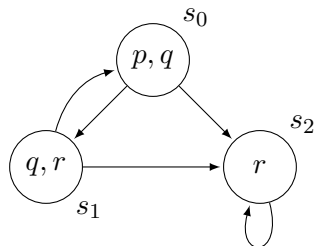
...

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

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

1. 应用

1.2 Linear-time temporal Logic (LTL) | Semantics

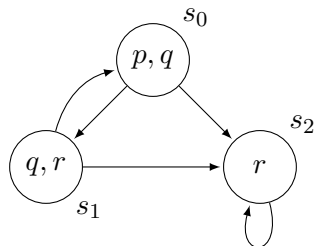


- $\mathcal{M}, s_0 \models p \wedge q$ holds
- $\mathcal{M}, s_0 \models \neg r$ holds
- $\mathcal{M}, s_0 \models \top$ holds
- $\mathcal{M}, s_0 \models X r$ holds
- $\mathcal{M}, s_0 \models X (q \wedge r)$ does not hold
- $\mathcal{M}, s_0 \models G \neg(p \wedge r)$ holds
- $\mathcal{M}, s_2 \models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

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

1. 应用

1.2 Linear-time temporal Logic (LTL) | Semantics

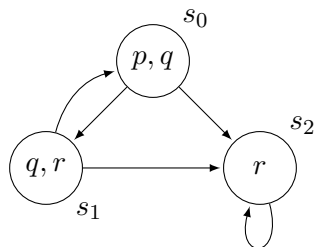


- $\mathcal{M}, s_0 \models p \wedge q$ holds
- $\mathcal{M}, s_0 \models \neg r$ holds
- $\mathcal{M}, s_0 \models \top$ holds
- $\mathcal{M}, s_0 \models X r$ holds
- $\mathcal{M}, s_0 \models X (q \wedge r)$ does not hold
- $\mathcal{M}, s_0 \models G \neg(p \wedge r)$ holds
- $\mathcal{M}, s_2 \models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

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

1. 应用

1.2 Linear-time temporal Logic (LTL) | Semantics

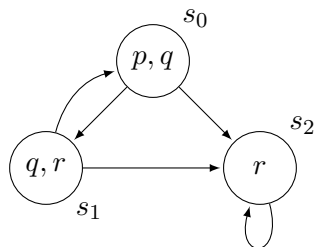


- $\mathcal{M}, s_0 \models p \wedge q$ holds
- $\mathcal{M}, s_0 \models \neg r$ holds
- $\mathcal{M}, s_0 \models \top$ holds
- $\mathcal{M}, s_0 \models X r$ holds
- $\mathcal{M}, s_0 \models X (q \wedge r)$ does not hold
- $\mathcal{M}, s_0 \models G \neg(p \wedge r)$ holds
- $\mathcal{M}, s_2 \models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

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

1. 应用

1.2 Linear-time temporal Logic (LTL) | Semantics

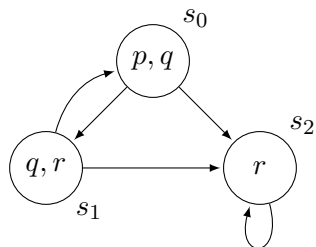


- $\mathcal{M}, s_0 \models p \wedge q$ holds
- $\mathcal{M}, s_0 \models \neg r$ holds
- $\mathcal{M}, s_0 \models \top$ holds
- $\mathcal{M}, s_0 \models X r$ holds
- $\mathcal{M}, s_0 \models X (q \wedge r)$ does not hold
- $\mathcal{M}, s_0 \models G \neg(p \wedge r)$ holds
- $\mathcal{M}, s_2 \models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

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

1. 应用

1.2 Linear-time temporal Logic (LTL) | Semantics

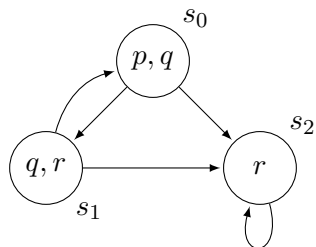


- $\mathcal{M}, s_0 \models p \wedge q$ holds
- $\mathcal{M}, s_0 \models \neg r$ holds
- $\mathcal{M}, s_0 \models \top$ holds
- $\mathcal{M}, s_0 \models X r$ holds
- $\mathcal{M}, s_0 \models X (q \wedge r)$ does not hold
- $\mathcal{M}, s_0 \models G \neg(p \wedge r)$ holds
- $\mathcal{M}, s_2 \models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

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

1. 应用

1.2 Linear-time temporal Logic (LTL) | Semantics

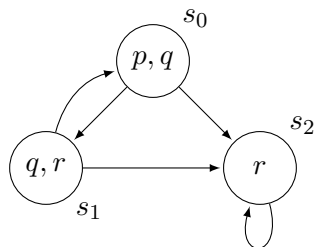


- $\mathcal{M}, s_0 \models p \wedge q$ holds
- $\mathcal{M}, s_0 \models \neg r$ holds
- $\mathcal{M}, s_0 \models \top$ holds
- $\mathcal{M}, s_0 \models X r$ holds
- $\mathcal{M}, s_0 \models X (q \wedge r)$ does not hold
- $\mathcal{M}, s_0 \models G \neg(p \wedge r)$ holds
- $\mathcal{M}, s_2 \models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

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

1. 应用

1.2 Linear-time temporal Logic (LTL) | Semantics

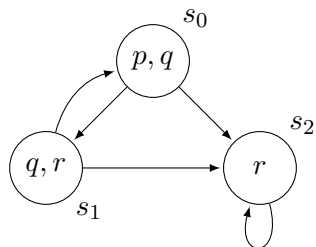


- $\mathcal{M}, s_0 \models p \wedge q$ holds
- $\mathcal{M}, s_0 \models \neg r$ holds
- $\mathcal{M}, s_0 \models \top$ holds
- $\mathcal{M}, s_0 \models X r$ holds
- $\mathcal{M}, s_0 \models X (q \wedge r)$ does not hold
- $\mathcal{M}, s_0 \models G \neg(p \wedge r)$ holds
- $\mathcal{M}, s_2 \models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

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

1. 应用

1.2 Linear-time temporal Logic (LTL) | Semantics

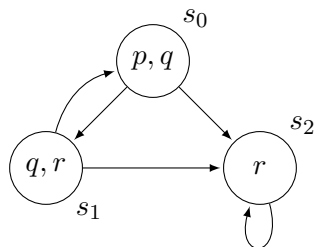


- $\mathcal{M}, s_0 \models p \wedge q$ holds
- $\mathcal{M}, s_0 \models \neg r$ holds
- $\mathcal{M}, s_0 \models \top$ holds
- $\mathcal{M}, s_0 \models X r$ holds
- $\mathcal{M}, s_0 \models X (q \wedge r)$ does not hold
- $\mathcal{M}, s_0 \models G \neg(p \wedge r)$ holds
- $\mathcal{M}, s_2 \models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

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

1. 应用

1.2 Linear-time temporal Logic (LTL) | Semantics

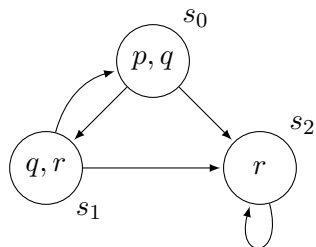


- $\mathcal{M}, s_0 \models p \wedge q$ holds
- $\mathcal{M}, s_0 \models \neg r$ holds
- $\mathcal{M}, s_0 \models \top$ holds
- $\mathcal{M}, s_0 \models X r$ holds
- $\mathcal{M}, s_0 \models X (q \wedge r)$ does not hold
- $\mathcal{M}, s_0 \models G \neg(p \wedge r)$ holds
- $\mathcal{M}, s_2 \models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

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

1. 应用

1.2 Linear-time temporal Logic (LTL) | Semantics

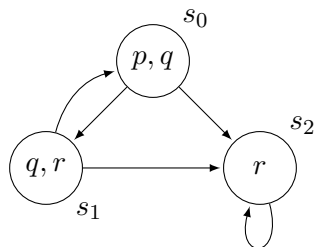


- $\mathcal{M}, s_0 \models p \wedge q$ holds
- $\mathcal{M}, s_0 \models \neg r$ holds
- $\mathcal{M}, s_0 \models \top$ holds
- $\mathcal{M}, s_0 \models X r$ holds
- $\mathcal{M}, s_0 \models X (q \wedge r)$ does not hold
- $\mathcal{M}, s_0 \models G \neg(p \wedge r)$ holds
- $\mathcal{M}, s_2 \models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

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

1. 应用

1.2 Linear-time temporal Logic (LTL) | Semantics

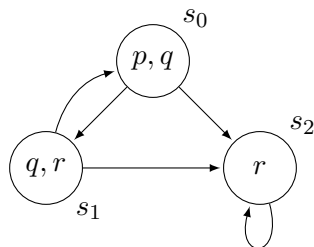


- $\mathcal{M}, s_0 \models p \wedge q$ holds
- $\mathcal{M}, s_0 \models \neg r$ holds
- $\mathcal{M}, s_0 \models \top$ holds
- $\mathcal{M}, s_0 \models X r$ holds
- $\mathcal{M}, s_0 \models X (q \wedge r)$ does not hold
- $\mathcal{M}, s_0 \models G \neg(p \wedge r)$ holds
- $\mathcal{M}, s_2 \models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

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

1. 应用

1.2 Linear-time temporal Logic (LTL) | Semantics

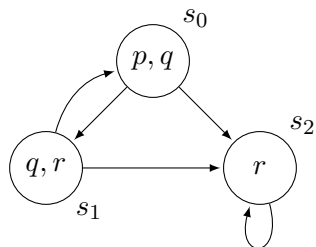


- $\mathcal{M}, s_0 \models p \wedge q$ holds
- $\mathcal{M}, s_0 \models \neg r$ holds
- $\mathcal{M}, s_0 \models \top$ holds
- $\mathcal{M}, s_0 \models X r$ holds
- $\mathcal{M}, s_0 \models X (q \wedge r)$ does not hold
- $\mathcal{M}, s_0 \models G \neg(p \wedge r)$ holds
- $\mathcal{M}, s_2 \models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

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

1. 应用

1.2 Linear-time temporal Logic (LTL) | Semantics

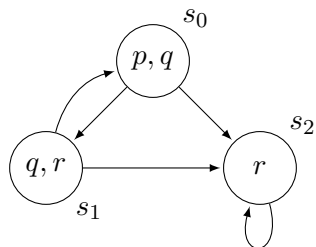


- $\mathcal{M}, s_0 \models p \wedge q$ holds
- $\mathcal{M}, s_0 \models \neg r$ holds
- $\mathcal{M}, s_0 \models \top$ holds
- $\mathcal{M}, s_0 \models X r$ holds
- $\mathcal{M}, s_0 \models X (q \wedge r)$ does not hold
- $\mathcal{M}, s_0 \models G \neg(p \wedge r)$ holds
- $\mathcal{M}, s_2 \models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

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

1. 应用

1.2 Linear-time temporal Logic (LTL) | Semantics

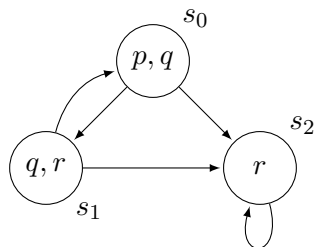


- $\mathcal{M}, s_0 \models p \wedge q$ holds
- $\mathcal{M}, s_0 \models \neg r$ holds
- $\mathcal{M}, s_0 \models \top$ holds
- $\mathcal{M}, s_0 \models X r$ holds
- $\mathcal{M}, s_0 \models X (q \wedge r)$ does not hold
- $\mathcal{M}, s_0 \models G \neg(p \wedge r)$ holds
- $\mathcal{M}, s_2 \models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

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

1. 应用

1.2 Linear-time temporal Logic (LTL) | Semantics

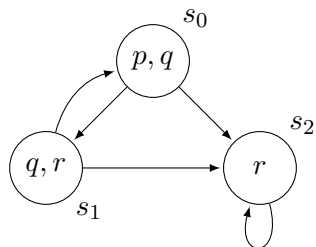


- $\mathcal{M}, s_0 \models p \wedge q$ holds
- $\mathcal{M}, s_0 \models \neg r$ holds
- $\mathcal{M}, s_0 \models \top$ holds
- $\mathcal{M}, s_0 \models X r$ holds
- $\mathcal{M}, s_0 \models X (q \wedge r)$ does not hold
- $\mathcal{M}, s_0 \models G \neg(p \wedge r)$ holds
- $\mathcal{M}, s_2 \models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

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

1. 应用

1.2 Linear-time temporal Logic (LTL) | Semantics

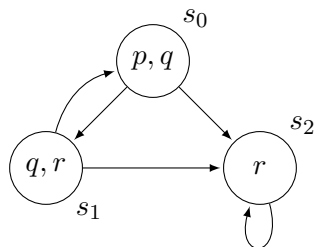


- $\mathcal{M}, s_0 \models p \wedge q$ holds
- $\mathcal{M}, s_0 \models \neg r$ holds
- $\mathcal{M}, s_0 \models \top$ holds
- $\mathcal{M}, s_0 \models X r$ holds
- $\mathcal{M}, s_0 \models X (q \wedge r)$ does not hold
- $\mathcal{M}, s_0 \models G \neg(p \wedge r)$ holds
- $\mathcal{M}, s_2 \models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

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

1. 应用

1.2 Linear-time temporal Logic (LTL) | Semantics

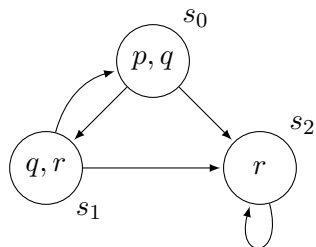


- $\mathcal{M}, s_0 \models p \wedge q$ holds
- $\mathcal{M}, s_0 \models \neg r$ holds
- $\mathcal{M}, s_0 \models \top$ holds
- $\mathcal{M}, s_0 \models X r$ holds
- $\mathcal{M}, s_0 \models X (q \wedge r)$ does not hold
- $\mathcal{M}, s_0 \models G \neg(p \wedge r)$ holds
- $\mathcal{M}, s_2 \models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

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

1. 应用

1.2 Linear-time temporal Logic (LTL) | Semantics

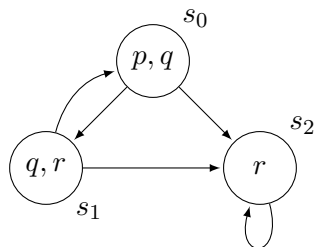


- $\mathcal{M}, s_0 \models p \wedge q$ holds
- $\mathcal{M}, s_0 \models \neg r$ holds
- $\mathcal{M}, s_0 \models \top$ holds
- $\mathcal{M}, s_0 \models X r$ holds
- $\mathcal{M}, s_0 \models X (q \wedge r)$ does not hold
- $\mathcal{M}, s_0 \models G \neg(p \wedge r)$ holds
- $\mathcal{M}, s_2 \models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

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

1. 应用

1.2 Linear-time temporal Logic (LTL) | Semantics

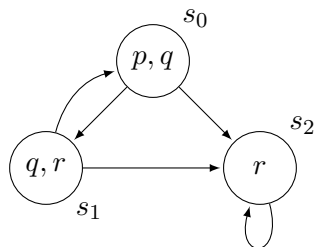


- $\mathcal{M}, s_0 \models p \wedge q$ holds
- $\mathcal{M}, s_0 \models \neg r$ holds
- $\mathcal{M}, s_0 \models \top$ holds
- $\mathcal{M}, s_0 \models X r$ holds
- $\mathcal{M}, s_0 \models X (q \wedge r)$ does not hold
- $\mathcal{M}, s_0 \models G \neg(p \wedge r)$ holds
- $\mathcal{M}, s_2 \models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

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

1. 应用

1.2 Linear-time temporal Logic (LTL) | Semantics

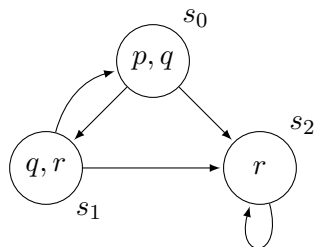


- $\mathcal{M}, s_0 \models p \wedge q$ holds
- $\mathcal{M}, s_0 \models \neg r$ holds
- $\mathcal{M}, s_0 \models \top$ holds
- $\mathcal{M}, s_0 \models X r$ holds
- $\mathcal{M}, s_0 \models X (q \wedge r)$ does not hold
- $\mathcal{M}, s_0 \models G \neg(p \wedge r)$ holds
- $\mathcal{M}, s_2 \models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

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

1. 应用

1.2 Linear-time temporal Logic (LTL) | Semantics

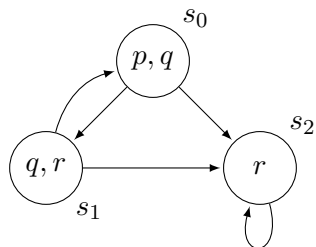


- $\mathcal{M}, s_0 \models p \wedge q$ holds
- $\mathcal{M}, s_0 \models \neg r$ holds
- $\mathcal{M}, s_0 \models \top$ holds
- $\mathcal{M}, s_0 \models X r$ holds
- $\mathcal{M}, s_0 \models X (q \wedge r)$ does not hold
- $\mathcal{M}, s_0 \models G \neg(p \wedge r)$ holds
- $\mathcal{M}, s_2 \models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

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

1. 应用

1.2 Linear-time temporal Logic (LTL) | Semantics

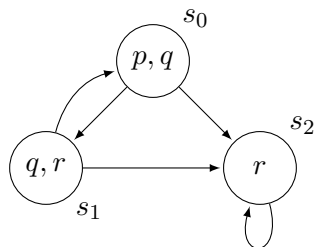


- $\mathcal{M}, s_0 \models p \wedge q$ holds
- $\mathcal{M}, s_0 \models \neg r$ holds
- $\mathcal{M}, s_0 \models \top$ holds
- $\mathcal{M}, s_0 \models X r$ holds
- $\mathcal{M}, s_0 \models X (q \wedge r)$ does not hold
- $\mathcal{M}, s_0 \models G \neg(p \wedge r)$ holds
- $\mathcal{M}, s_2 \models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

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

1. 应用

1.2 Linear-time temporal Logic (LTL) | Semantics

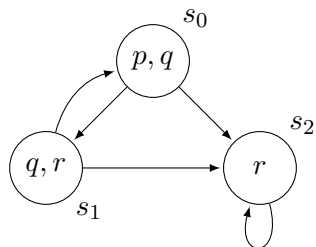


- $\mathcal{M}, s_0 \models p \wedge q$ holds
- $\mathcal{M}, s_0 \models \neg r$ holds
- $\mathcal{M}, s_0 \models \top$ holds
- $\mathcal{M}, s_0 \models X r$ holds
- $\mathcal{M}, s_0 \models X (q \wedge r)$ does not hold
- $\mathcal{M}, s_0 \models G \neg(p \wedge r)$ holds
- $\mathcal{M}, s_2 \models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

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

1. 应用

1.2 Linear-time temporal Logic (LTL) | Semantics



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

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

1. 应用

1.2 LTL | Practical patterns of Specification

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:

$$G F \text{ enabled}$$

1. 应用

1.2 LTL | Practical patterns of Specification

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:

$$G F \text{ enabled}$$

1. 应用

1.2 LTL | Practical patterns of Specification

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:

$$G F \text{ enabled}$$

1. 应用

1.2 LTL | Practical patterns of Specification

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:

$$G F \text{ enabled}$$

1. 应用

1.2 LTL | Practical patterns of Specification

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*:

$$F G \text{ deadlock}$$

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

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

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

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

1. 应用

1.2 LTL | Practical patterns of Specification

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*:

$F G \text{ deadlock}$

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

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

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

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

1. 应用

1.2 LTL | Practical patterns of Specification

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*:

$$F G \text{ deadlock}$$

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

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

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

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

1. 应用

1.2 LTL | Practical patterns of Specification

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*:

$$F G \text{ deadlock}$$

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

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

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

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

1. 应用

1.2 LTL | Practical patterns of Specification

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

1. 应用

1.2 LTL | Practical patterns of Specification

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

1. 应用

1.2 LTL | Practical patterns of Specification

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

1. 应用

1.2 LTL | Practical patterns of Specification

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

1. 应用

1.2 LTL | Practical patterns of Specification

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

1. 应用

1.2 LTL | Practical patterns of Specification

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

1. 应用

1.2 LTL | Equivalences

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

$$\neg G \phi \equiv F \neg\phi \quad \neg F \phi \equiv G \neg\phi \quad \neg X \phi \equiv X \neg\phi$$

$$\neg(\phi U \psi) \equiv \neg\phi R \neg\psi \quad \neg(\phi R \psi) \equiv \neg\phi U \neg\psi$$

$$F(\phi \vee \psi) \equiv F\phi \vee F\psi \quad G(\phi \wedge \psi) \equiv G\phi \wedge G\psi$$

$$F\phi \equiv \top U \phi \quad G\phi \equiv \perp R \phi$$

$$\phi U \psi \equiv \phi W \psi \wedge F\psi \quad \phi W \psi \equiv \phi U \psi \vee G\psi$$

$$\phi W \psi \equiv \psi R(\phi \vee \psi) \quad \phi R \psi \equiv \psi W(\phi \wedge \psi)$$

1. 应用

1.3 Computation Tree Logic (CTL)

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

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

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

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:

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

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

1. 应用

1.3 Computation Tree Logic (CTL)

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

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

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

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:

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

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

1. 应用

1.3 Computation Tree Logic (CTL) | Semantics

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 $\mathcal{M}, s \not\models \perp$
- 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 $\mathcal{M}, s \models \phi_1$

1. 应用

1.3 Computation Tree Logic (CTL) | Semantics

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

- ⑧ $\mathcal{M}, s \models AX \phi$ iff *for all* s_1 such that $s \rightarrow s_1$ we have $\mathcal{M}, s_1 \models \phi$
- ⑨ $\mathcal{M}, s \models EX \phi$ iff *for some* s_1 such that $s \rightarrow s_1$, we have $\mathcal{M}, s_1 \models \phi$
- ⑩ $\mathcal{M}, s \models AG \phi$ iff *for all paths* $s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow \dots$, where s_1 equals s , and *for all* s_i along the path, we have $\mathcal{M}, s_i \models \phi$
- ⑪ $\mathcal{M}, s \models EG \phi$ iff *there is a path* $s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow \dots$, where s_1 equals s , and *for all* s_i along the path, we have $\mathcal{M}, s_i \models \phi$
- ⑫ $\mathcal{M}, s \models AF \phi$ iff *for all paths* $s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow \dots$, where s_1 equals s , and *there is* some s_i such that $\mathcal{M}, s_i \models \phi$
- ⑬ $\mathcal{M}, s \models EF \phi$ iff *there is a path* $s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow \dots$, where s_1 equals s , and *there is* some s_i such that $\mathcal{M}, s_i \models \phi$

1. 应用

1.3 Computation Tree Logic (CTL) | Semantics

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

- 14 $\mathcal{M}, s \models A[\phi_1 \cup \phi_2]$ iff *for all paths* $s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow \dots$, where s_1 equals s , that path satisfies $\phi_1 \cup \phi_2$, i.e., there is some s_i along the path, such that $\mathcal{M}, s_i \models \phi_2$, and, for each $j < i$, we have $\mathcal{M}, s_j \models \phi_1$
- 15 $\mathcal{M}, s \models E[\phi_1 \cup \phi_2]$ iff *there is a path* $s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow \dots$, where s_1 equals s , that path satisfies $\phi_1 \cup \phi_2$, i.e., there is some s_i along the path, such that $\mathcal{M}, s_i \models \phi_2$, and, for each $j < i$, we have $\mathcal{M}, s_j \models \phi_1$

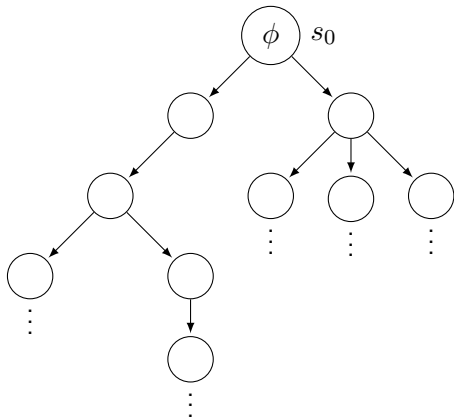
1. 应用

1.3 Computation Tree Logic (CTL) | Semantics

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

For example,

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



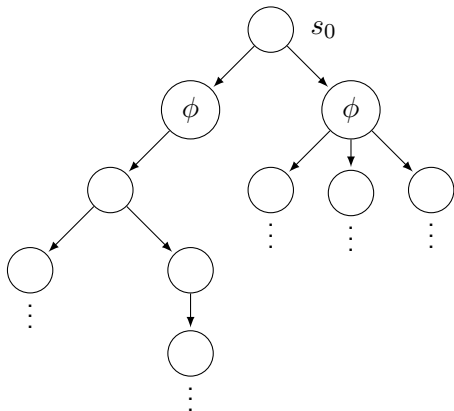
1. 应用

1.3 Computation Tree Logic (CTL) | Semantics

$\mathcal{M}, s \models AX \phi$ iff *for all* s_1
such that $s \rightarrow s_1$ we have
 $\mathcal{M}, s_1 \models \phi$

For example,

$$\mathcal{M}, s_0 \models AX \phi$$



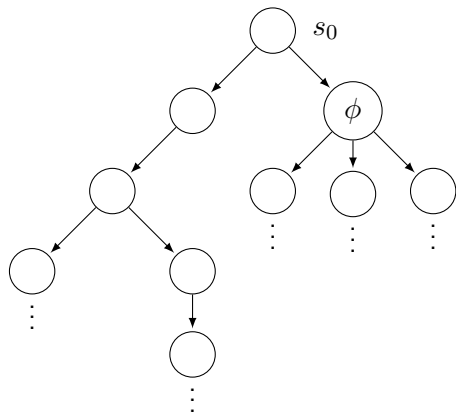
1. 应用

1.3 Computation Tree Logic (CTL) | Semantics

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

For example,

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



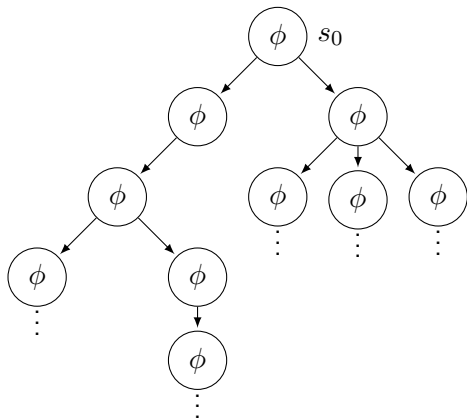
1. 应用

1.3 Computation Tree Logic (CTL) | Semantics

$\mathcal{M}, s \models \text{AG } \phi$ iff *for all paths*
 $s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow \dots$, where
 s_1 equals s , and *for all* s_i
along the path, we have
 $\mathcal{M}, s_i \models \phi$

For example,

$$\mathcal{M}, s_0 \models \text{AG } \phi$$



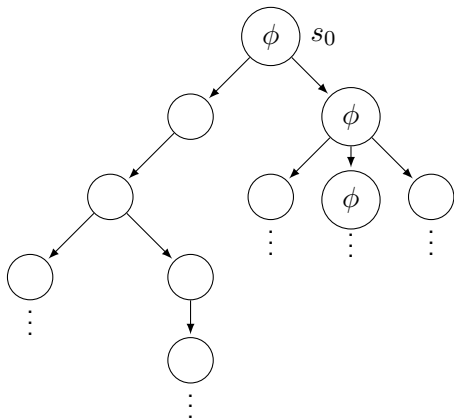
1. 应用

1.3 Computation Tree Logic (CTL) | Semantics

$\mathcal{M}, s \models \text{EG } \phi$ iff *there is a path* $s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow \dots$, where s_1 equals s , and *for all* s_i along the path, we have $\mathcal{M}, s_i \models \phi$

For example,

$$\mathcal{M}, s_0 \models \text{EG } \phi$$



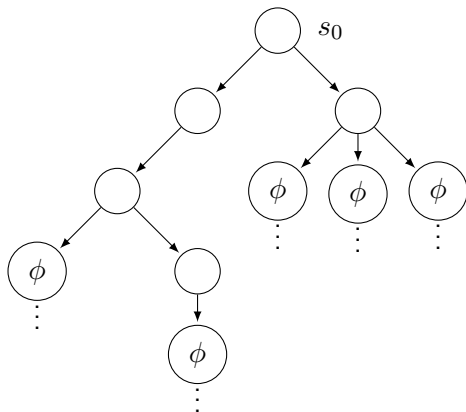
1. 应用

1.3 Computation Tree Logic (CTL) | Semantics

$\mathcal{M}, s \models \text{AF } \phi$ iff *for all* paths $s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow \dots$, where s_1 equals s , and *there is* some s_i such that $\mathcal{M}, s_i \models \phi$

For example,

$$\mathcal{M}, s_0 \models \text{AF } \phi$$



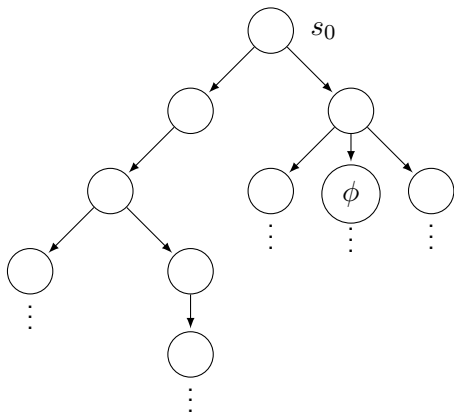
1. 应用

1.3 Computation Tree Logic (CTL) | Semantics

$\mathcal{M}, s \models \text{EF } \phi$ iff *there is a path* $s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow \dots$, where s_1 equals s , and *there is* some s_i such that $\mathcal{M}, s_i \models \phi$

For example,

$$\mathcal{M}, s_0 \models \text{EF } \phi$$



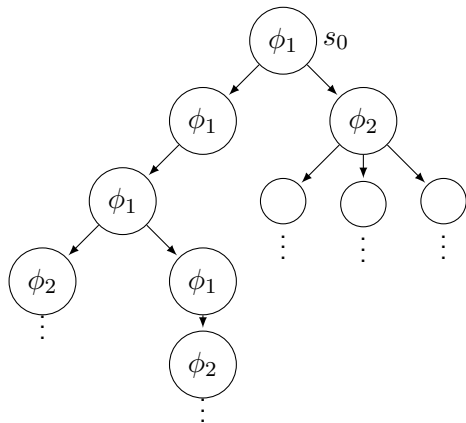
1. 应用

1.3 Computation Tree Logic (CTL) | Semantics

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

For example,

$$\mathcal{M}, s_0 \models A[\phi_1 \text{ U } \phi_2]$$



1. 应用

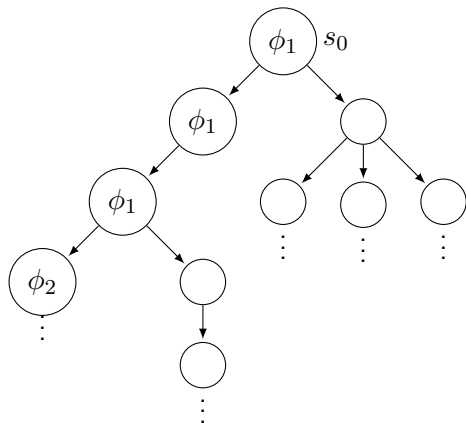
1.3 Computation Tree Logic (CTL) | Semantics

$\mathcal{M}, s \models E[\phi_1 \text{ U } \phi_2]$ iff *there is a path*

$s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow \dots$, where s_1 equals s , that path satisfies $\phi_1 \text{ U } \phi_2$, i.e., there is some s_i along the path, such that $\mathcal{M}, s_i \models \phi_2$, and, for each $j < i$, we have $\mathcal{M}, s_j \models \phi_1$

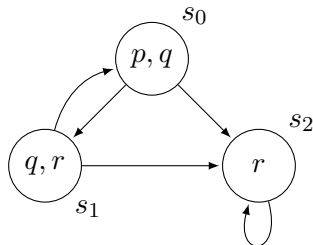
For example,

$$\mathcal{M}, s_0 \models E[\phi_1 \text{ U } \phi_2]$$



1. 应用

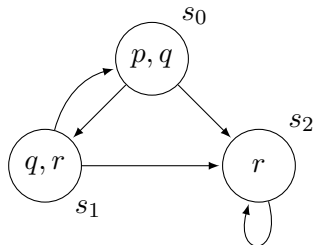
1.3 Computation Tree Logic (CTL) | Semantics



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

1. 应用

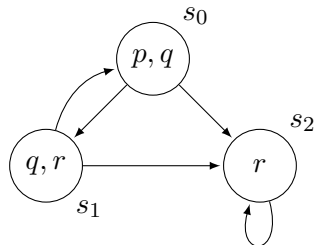
1.3 Computation Tree Logic (CTL) | Semantics



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

1. 应用

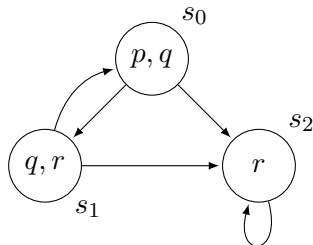
1.3 Computation Tree Logic (CTL) | Semantics



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

1. 应用

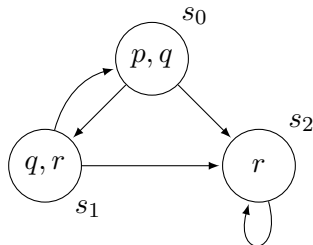
1.3 Computation Tree Logic (CTL) | Semantics



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

1. 应用

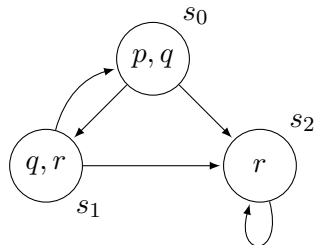
1.3 Computation Tree Logic (CTL) | Semantics



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

1. 应用

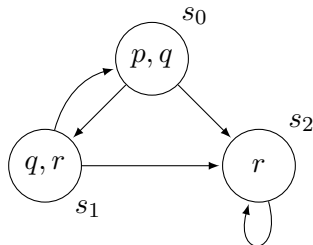
1.3 Computation Tree Logic (CTL) | Semantics



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

1. 应用

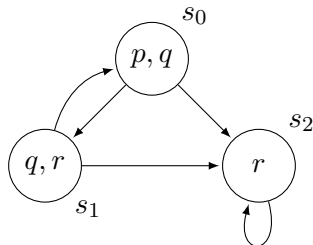
1.3 Computation Tree Logic (CTL) | Semantics



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

1. 应用

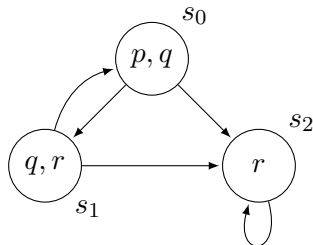
1.3 Computation Tree Logic (CTL) | Semantics



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

1. 应用

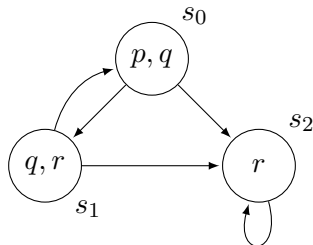
1.3 Computation Tree Logic (CTL) | Semantics



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

1. 应用

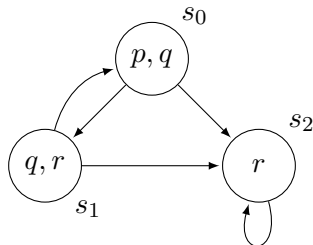
1.3 Computation Tree Logic (CTL) | Semantics



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

1. 应用

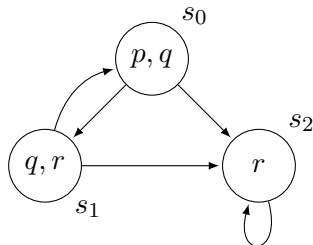
1.3 Computation Tree Logic (CTL) | Semantics



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

1. 应用

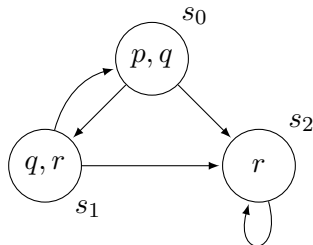
1.3 Computation Tree Logic (CTL) | Semantics



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

1. 应用

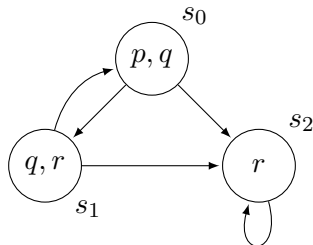
1.3 Computation Tree Logic (CTL) | Semantics



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

1. 应用

1.3 Computation Tree Logic (CTL) | Semantics



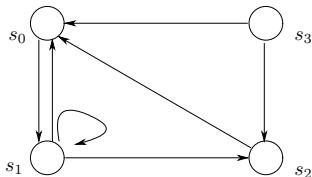
- $\mathcal{M}, s_0 \models p \wedge q$ holds
- $\mathcal{M}, s_0 \models \neg r$ holds
- $\mathcal{M}, s_0 \models \top$ holds
- $\mathcal{M}, s_0 \models \text{EX } (q \wedge r)$ holds
- $\mathcal{M}, s_0 \models \neg \text{AX } (q \wedge r)$ holds
- $\mathcal{M}, s_0 \models \neg \text{EF } (p \wedge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

1. 应用

1.3 Computation Tree Logic (CTL) | Semantics

回顾: 反例:

Given a set of states $A = \{s_0, s_1, s_2, s_3\}$, let R^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 $(s, s') \in R^M$.



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' ?

1. 应用

1.3 Computation Tree Logic (CTL) | Semantics

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

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

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

1. 应用

1.3 Computation Tree Logic (CTL) | Semantics

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

$$C_1 \stackrel{\text{def}}{=} P(x, x)$$

$$C_2 \stackrel{\text{def}}{=} P(x, y) \wedge P(y, z) \rightarrow P(x, z)$$

$$C_3 \stackrel{\text{def}}{=} P(u, v) \rightarrow \perp$$

$$C_4 \stackrel{\text{def}}{=} R(x, y) \rightarrow P(x, y)$$

1. 应用

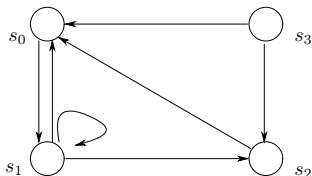
1.3 Computation Tree Logic (CTL) | Semantics

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

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



1. 应用

1.3 Computation Tree Logic (CTL) | Equivalences

$$\neg AF \phi \equiv EG \neg\phi$$

$$\neg EF \phi \equiv AG \neg\phi$$

$$\neg AX \phi \equiv EX \neg\phi$$

$$AF \phi \equiv A[\top U \phi]$$

$$EF \phi \equiv E[\top U \phi]$$

1. 应用

1.3 Computation Tree Logic (CTL) | LTL v.s. CTL

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:

$$FG p$$

How to express it in CTL? $AFAG p$? *No*

Another LTL formula?

$$F p \rightarrow F q$$

怎么办?

1. 应用

1.3 Computation Tree Logic (CTL) | LTL v.s. CTL

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:

$$FG p$$

How to express it in CTL? $AFAG p$? *No*

Another LTL formula?

$$F p \rightarrow F q$$

怎么办?

1. 应用

1.3 Computation Tree Logic (CTL) | LTL v.s. CTL

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:

$$FG p$$

How to express it in CTL? $AFAG p$? *No*

Another LTL formula?

$$F p \rightarrow F q$$

怎么办?

1. 应用

1.3 Computation Tree Logic (CTL) | LTL v.s. CTL

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:

$$FG p$$

How to express it in CTL? $AFAG p$? *No*

Another LTL formula?

$$F p \rightarrow F q$$

怎么办?

1. 应用

1.3 Computation Tree Logic (CTL) | LTL v.s. CTL

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:

$$FG p$$

How to express it in CTL? $AFAG p$? *No*

Another LTL formula?

$$F p \rightarrow F q$$

怎么办?

1. 应用

1.3 Computation Tree Logic (CTL) | LTL v.s. CTL

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:

$$FG p$$

How to express it in CTL? $AFAG p$? *No*

Another LTL formula?

$$F p \rightarrow F q$$

怎么办?

1. 应用

1.3 Computation Tree Logic (CTL) | LTL v.s. CTL

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:

$$FG p$$

How to express it in CTL? $AFAG p$? *No*

Another LTL formula?

$$F p \rightarrow F q$$

怎么办?

1. 应用

1.3 Computation Tree Logic (CTL) | LTL v.s. CTL

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:

$$FG p$$

How to express it in CTL? $AFAG p$? *No*

Another LTL formula?

$$F p \rightarrow F q$$

怎么办?

1. 应用

1.3 Computation Tree Logic (CTL) | LTL v.s. CTL | CTL*

*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:

- $A[(p \text{ U } r) \vee (q \text{ U } r)]$: 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.

1. 应用

1.3 Computation Tree Logic (CTL) | LTL v.s. CTL | CTL*

*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:

- $A[(p \text{ U } r) \vee (q \text{ U } r)]$: 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.

1. 应用

1.3 Computation Tree Logic (CTL) | LTL v.s. CTL | CTL*

*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:

- $A[(p \text{ U } r) \vee (q \text{ U } r)]$: 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.

1. 应用

1.3 Computation Tree Logic (CTL) | LTL v.s. CTL | CTL*

*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:

- $A[(p \text{ U } r) \vee (q \text{ U } r)]$: 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.

1. 应用

1.3 Computation Tree Logic (CTL) | Semantics

定义: CTL*

The syntax of CTL* involves two classes of formulas:

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

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

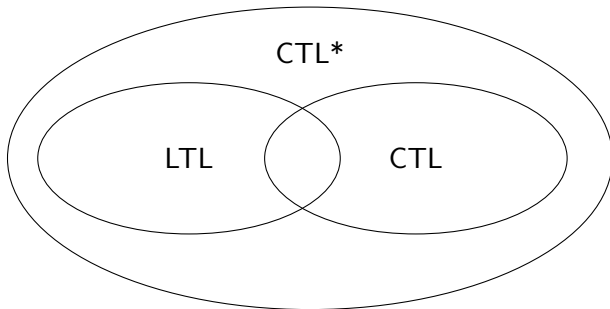
where p is any atomic formula and α any path formula

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

$$\alpha ::= \phi \mid (\neg\alpha) \mid (\alpha \wedge \alpha) \mid (\alpha \cup \alpha) \mid (G \alpha) \mid (F \alpha) \mid (X \alpha)$$

1. 应用

1.3 Computation Tree Logic (CTL) | Semantics



1. 应用

1.3 Computation Tree Logic (CTL) | Semantics

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)

1. 应用

1.3 Computation Tree Logic (CTL) | Semantics

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)

1. 应用

1.3 Computation Tree Logic (CTL) | Semantics

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)

1. 应用

1.3 Computation Tree Logic (CTL) | Semantics

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)

1. 应用

1.3 Computation Tree Logic (CTL) | Semantics

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)

1. 应用

1.3 Computation Tree Logic (CTL) | Semantics

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)

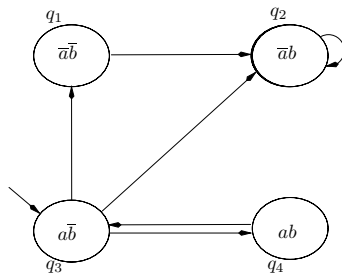


Figure 3.39. A model \mathcal{M} .

2. Consider the system of Figure 3.39. For each of the formulas ϕ :

- $G a$
- $a U b$
- $a U X (a \wedge \neg b)$
- $X \neg b \wedge G (\neg a \vee \neg b)$
- $X (a \wedge b) \wedge F (\neg a \wedge \neg b)$
 - Find a path from the initial state q_3 which satisfies ϕ .
 - Determine whether $\mathcal{M}, q_3 \models \phi$.

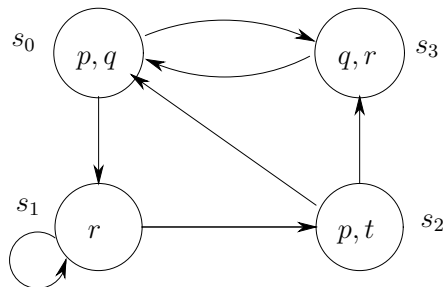


Figure 3.41. Another model with four states.

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