Specifications in Temporal Logic

Temporal Logic: A Class of Modal Logics

- Modal Logic: alternative notions of truth like *is it possible/necessary that* φ *is true?*

Temporal Logic: A Class of Modal Logics

- Modal Logic: alternative notions of truth like *is it possible/necessary that φ is true?*
- In modal logic interpretations are defined as Kripke structures, i.e., a set of worlds W and an accessibility relation R in $W \times W$

Temporal Logic: A Class of Modal Logics

- Modal Logic: alternative notions of truth like *is it possible/necessary that φ is true?*
- In modal logic interpretations are defined as Kripke structures, i.e., a set of worlds W and an accessibility relation R in $W \times W$
- Propositions are interpreted in each world

Temporal Logic: A Class of Modal Logics

- Modal Logic: alternative notions of truth like *is it possible/necessary that* φ *is true?*
- In modal logic interpretations are defined as Kripke structures, i.e., a set of worlds W and an accessibility relation R in $W \times W$
- Propositions are interpreted in each world
- Modalities quantify over the set of words accessible from the current one via R

Temporal Logic: A Class of Modal Logics

- Modal Logic: alternative notions of truth like *is it possible/necessary that* φ *is true?*
- In modal logic interpretations are defined as Kripke structures, i.e., a set of worlds W and an accessibility relation R in $W \times W$
- Propositions are interpreted in each world
- Modalities quantify over the set of words accessible from the current one via R
- A specific modal logic is characterized by the properties of R (reflexivity, transitivity, etc)

Temporal Logic

- Temporal logic is a special type of modal logic in which the truth of a formula depends on the time in which it is evaluated
- Typical temporal operators are
  - Eventually Φ: in some future instant Φ is true
  - Always Φ: in all future instants Φ is true

Several Types of Temporal Logics

- Linear Temporal Logic (LTL) is linear in the future; properties are defined on a path

Several Types of Temporal Logics

- Linear Temporal Logic (LTL) is linear in the future; properties are defined on a path
- Computational Tree Logic (CTL) is branching in the future; properties are defined on a tree

Several Types of Temporal Logics

- Linear Temporal Logic (LTL) is linear in the future; properties are defined on a path
- Computational Tree Logic (CTL) is branching in the future; properties are defined on a tree
- LTL and CTL are incomparable logics: There exist formulas in one logic that are not expressible in the other

Several Types of Temporal Logics

- Linear Temporal Logic (LTL) is linear in the future; properties are defined on a path
- Computational Tree Logic (CTL) is branching in the future; properties are defined on a tree
- LTL and CTL are incomparable logics: There exist formulas in one logic that are not expressible in the other

Several Types of Temporal Logics

- Linear Temporal Logic (LTL) is linear in the future; properties are defined on a path
- Computational Tree Logic (CTL) is branching in the future; properties are defined on a tree
- LTL and CTL are incomparable logics: There exist formulas in one logic that are not expressible in the other
- LTL and CTL are submsumed by CTL*, which in turn, is subsumed by the μ-calculus (a fixpoint logic)

Model Checking Problem

- Fixed a (Kripke) model M (a transition system), an initial state s_0, and a temporal property φ

$$M, s_0 \models \varphi?$$

where $\models$ = satisfiability relation

Computation Tree Logic

Kripke Models and Branching Time

- In CTL (Computation Tree Logic) time is branching in the future, i.e., in a Kripke Model a world has a set of possible successors
- If we unfold the model we obtain an infinite tree; for each node (world/state) of the tree we specify which propositions are true and which are false
- CTL temporal operators quantify over paths and states of the computation tree

(Propositional) Computation Tree Logic: (P)CTL

- Atomic propositions: $p, q, r, \ldots$

(Propositional) Computation Tree Logic: (P)CTL

- Atomic propositions: $p, q, r, \ldots$
- Classical connectives: $\neg\psi \quad \varphi \wedge \psi \quad \varphi \vee \psi \quad \varphi \supset \psi$

(Propositional) Computation Tree Logic: (P)CTL

- Atomic propositions: $p, q, r, \ldots$
- Classical connectives: $\neg\psi \quad \varphi \wedge \psi \quad \varphi \vee \psi \quad \varphi \supset \psi$
- Two types of modalities:

(Propositional) Computation Tree Logic: (P)CTL

- Atomic propositions: $p, q, r, \ldots$
- Classical connectives: $\neg\psi \quad \varphi \wedge \psi \quad \varphi \vee \psi \quad \varphi \supset \psi$
- Two types of modalities:
  - Path quantifiers $\mathbb{P}$::= **E**, **A**

(Propositional) Computation Tree Logic: (P)CTL

- Atomic propositions: $p, q, r, \ldots$
- Classical connectives: $\neg\psi \quad \varphi \wedge \psi \quad \varphi \vee \psi \quad \varphi \supset \psi$
- Two types of modalities:
  - Path quantifiers $\mathbb{P} ::= \mathbf{E}, \mathbf{A}$
  - Temporal modalities $\mathbb{T} ::= \mathbf{X}, \mathbf{F}, \mathbf{G}, \mathbf{U}$

(Propositional) Computation Tree Logic: (P)CTL

- Atomic propositions: $p, q, r, \ldots$
- Classical connectives: $\neg\psi \quad \varphi \wedge \psi \quad \varphi \vee \psi \quad \varphi \supset \psi$
- Two types of modalities:
  - Path quantifiers $\mathbb{P} ::= \mathbf{E}, \mathbf{A}$
  - Temporal modalities $\mathbb{T} ::= \mathbf{X}, \mathbf{F}, \mathbf{G}, \mathbf{U}$
- CTL formulas have the form $\mathbb{P}\mathbb{T}\varphi$

CTL Modalities

- A formula with no top-level modality refers to the current state

CTL Modalities

- A formula with no top-level modality refers to the current state
- **A** (*for all paths*) and **E** (*there exists a path*) are always combined with **X** (next) and **U** (until)

CTL Modalities

- A formula with no top-level modality refers to the current state
- **A** (*for all paths*) and **E** (*there exists a path*) are always combined with **X** (next) and **U** (until)
- $\mathbf{EX}\varphi \;=\;$ exists a path s.t. next φ

CTL Modalities

- A formula with no top-level modality refers to the current state
- **A** (*for all paths*) and **E** (*there exists a path*) are always combined with **X** (next) and **U** (until)
- $\mathbf{EX}\varphi$ = exists a path s.t. next φ
- $\mathbf{EF}\varphi$ = exists a path s.t. eventually φ

CTL Modalities

- A formula with no top-level modality refers to the current state
- **A** (*for all paths*) and **E** (*there exists a path*) are always combined with **X** (next) and **U** (until)
- $\mathbf{EX}\varphi$ = exists a path s.t. next φ
- $\mathbf{EF}\varphi$ = exists a path s.t. eventually φ
- $\mathbf{EG}\varphi$ = exists a path s.t. always φ

CTL Modalities

- A formula with no top-level modality refers to the current state
- **A** (*for all paths*) and **E** (*there exists a path*) are always combined with **X** (next) and **U** (until)
- $\mathbf{EX}\varphi \;=\;$ exists a path s.t. next φ
- $\mathbf{EF}\varphi \;=\;$ exists a path s.t. eventually φ
- $\mathbf{EG}\varphi \;=\;$ exists a path s.t. always φ
- $\mathbf{E}(\varphi\mathbf{U}\psi) \;=\;$ exists a path s.t. φ until ψ

CTL Modalities

- A formula with no top-level modality refers to the current state
- **A** (*for all paths*) and **E** (*there exists a path*) are always combined with **X** (next) and **U** (until)
- $\mathbf{EX}\varphi \;=\;$ exists a path s.t. next φ
- $\mathbf{EF}\varphi \;=\;$ exists a path s.t. eventually φ
- $\mathbf{EG}\varphi \;=\;$ exists a path s.t. always φ
- $\mathbf{E}(\varphi\mathbf{U}\psi) \;=\;$ exists a path s.t. φ until ψ
- $\mathbf{A}(\varphi\mathbf{U}\psi) \;=\;$ for all paths φ until ψ

Other Connectives

- $\mathbf{EF}\varphi \equiv \mathbf{E}(\mathit{true}\ \mathbf{U}\ \varphi)$ potentially

Other Connectives

- $\mathbf{EF}\varphi \;\equiv\; \mathbf{E}(\mathit{true}\ \mathbf{U}\ \varphi)$ potentially
- $\mathbf{AF}\varphi \;=\; \mathbf{A}(\mathit{true}\ \mathbf{U}\ \varphi)$ inevitable

Other Connectives

- $\mathbf{EF}\varphi \;\equiv\; \mathbf{E}(\mathit{true}\ \mathbf{U}\ \varphi)$ potentially
- $\mathbf{AF}\varphi \;=\; \mathbf{A}(\mathit{true}\ \mathbf{U}\ \varphi)$ inevitable
- $\mathbf{AG}\varphi \;\equiv\; \neg\mathbf{EF}\neg\varphi$ invariantly

Other Connectives

- $\mathbf{EF}\varphi \;\equiv\; \mathbf{E}(\mathit{true}\ \mathbf{U}\ \varphi)$ potentially
- $\mathbf{AF}\varphi \;=\; \mathbf{A}(\mathit{true}\ \mathbf{U}\ \varphi)$ inevitable
- $\mathbf{AG}\varphi \;\equiv\; \neg\mathbf{EF}\neg\varphi$ invariantly
- $\mathbf{AX}\varphi \;=\; \mathbf{EX}\neg\varphi$ for all paths next

Example of formulas

- $\mathit{Started} \wedge \mathbf{EX}\,\mathit{Ready}$: Started holds in the current state, there exists a successor state in which Ready holds

Example of formulas

- $\mathit{Started} \land \mathbf{EX}\,\mathit{Ready}$: Started holds in the current state, there exists a successor state in which Ready holds
- $\mathbf{EF}(\mathit{Started} \land \neg \mathit{Ready})$: it is possible to get to a state where Started holds but Ready does not hold.

Example of formulas

- $Started \wedge \mathbf{EX} Ready$: Started holds in the current state, there exists a successor state in which Ready holds
- $\mathbf{EF}(Started \wedge \neg Ready)$: it is possible to get to a state where Started holds but Ready does not hold.
- $\mathbf{AG}(Req \supset \mathbf{AF} Ack)$: if a Request occurs, then it will be eventually acknowledged.

Example of formulas

- $\mathit{Started} \wedge \mathbf{EX}\mathit{Ready}$: Started holds in the current state, there exists a successor state in which Ready holds
- $\mathbf{EF}(\mathit{Started} \wedge \neg \mathit{Ready})$: it is possible to get to a state where Started holds but Ready does not hold.
- $\mathbf{AG}(\mathit{Req} \supset \mathbf{AF}\mathit{Ack})$: if a Request occurs, then it will be eventually acknowledged.
- $\mathbf{AG}(\mathbf{AF}\mathit{DeviceEnabled})$: DeviceEnabled holds infitely often on every computation path.

Example of formulas

- $\mathit{Started} \wedge \mathbf{EX}\mathit{Ready}$: Started holds in the current state, there exists a successor state in which Ready holds
- $\mathbf{EF}(\mathit{Started} \wedge \neg \mathit{Ready})$: it is possible to get to a state where Started holds but Ready does not hold.
- $\mathbf{AG}(\mathit{Req} \supset \mathbf{AF}\mathit{Ack})$: if a Request occurs, then it will be eventually acknowledged.
- $\mathbf{AG}(\mathbf{AF}\mathit{DeviceEnabled})$: DeviceEnabled holds infitely often on every computation path.
- $\mathbf{AG}(\mathbf{EF}\mathit{Restart})$: from any state it is possible to get to the Restart state.

Semantics

A CTL model is a triple $M = \langle S, R, L \rangle$ (Kripke model) where

- S is a non empty set of states

Semantics

A CTL model is a triple $M = \langle S, R, L \rangle$ (Kripke model) where

- S is a non empty set of states
- $R \subseteq S \to S$ is a total relation (branching in the future)
  R total means that for each $s \in S$ there exists at least one s' s.t. $\langle s, s' \rangle \in R$

Semantics

A CTL model is a triple $M = \langle S, R, L \rangle$ (Kripke model) where

- S is a non empty set of states
- $R \subseteq S \to S$ is a total relation (branching in the future)
  R total means that for each $s \in S$ there exists at least one s' s.t. $\langle s, s' \rangle \in R$
- $L : S \to 2^{AP}$ assigns to each state $s \in S$ the atomic formulas that are true in s

Paths

- A path σ is an infinite sequence of states $s_0 s_1 \ldots$ s.t. $\langle s_i, s_{i+1} \rangle \in R$

Paths

- A path σ is an infinite sequence of states $s_0 s_1 \ldots$ s.t. $\langle s_i, s_{i+1} \rangle \in R$
- $\sigma[i]$ identifies the i-th state in the sequence σ

Paths

- A path σ is an infinite sequence of states $s_0 s_1 \ldots$ s.t. $\langle s_i, s_{i+1} \rangle \in R$
- $\sigma[i]$ identifies the i-th state in the sequence σ
- The set of paths that start in s in M is

$$P_M(s) = \{\sigma \mid \sigma \text{ is a path s.t. } \sigma[0] = s\}$$

Paths

- A path σ is an infinite sequence of states $s_0 s_1 \ldots$ s.t. $\langle s_i, s_{i+1} \rangle \in R$
- $\sigma[i]$ identifies the i-th state in the sequence σ
- The set of paths that start in s in M is

$$P_M(s) = \{\sigma \mid \sigma \text{ is a path s.t. } \sigma[0] = s\}$$

- For each M there exists an infinite computation tree where each node is a state $s \in S$ and s.t. $\langle s', s'' \rangle$ is an edge iff $\langle s', s'' \rangle \in R$

Satisfiability

Fixed $M = \langle S, R, L \rangle$, the relation $M, s \models \varphi$ (M satisfies φ in s) is defined as

- $s \models p$ if $p \in L(s)$

$P_M(s)$ is the set of infinite paths from s in M

Satisfiability

Fixed $M = \langle S, R, L \rangle$, the relation $M, s \models \varphi$ (M satisfies φ in s) is defined as

- $s \models p$ if $p \in L(s)$
- $s \models \neg\phi$ if $s \not\models \phi$

$P_M(s)$ is the set of infinite paths from s in M

Satisfiability

Fixed $M = \langle S, R, L \rangle$, the relation $M, s \models \varphi$ (M satisfies φ in s) is defined as

- $s \models p$ if $p \in L(s)$
- $s \models \neg\phi$ if $s \not\models \phi$
- $s \models \phi \vee \psi$ if $s \models \phi$ or $s \models \psi$

$P_M(s)$ is the set of infinite paths from s in M

Satisfiability

Fixed $M = \langle S, R, L \rangle$, the relation $M, s \models \varphi$ (M satisfies φ in s) is defined as

- $s \models p$ if $p \in L(s)$
- $s \models \neg\phi$ if $s \not\models \phi$
- $s \models \phi \vee \psi$ if $s \models \phi$ or $s \models \psi$
- $s \models \mathbf{EX}\phi$ if $\exists \sigma \in P_M(s)$ t.c. $s[1] \models \phi$

$P_M(s)$ is the set of infinite paths from s in M

Satisfiability

Fixed $M = \langle S, R, L \rangle$, the relation $M, s \models \varphi$ (M satisfies φ in s) is defined as

- $s \models p$ if $p \in L(s)$
- $s \models \neg\phi$ if $s \not\models \phi$
- $s \models \phi \vee \psi$ if $s \models \phi$ or $s \models \psi$
- $s \models \mathbf{EX}\phi$ if $\exists \sigma \in P_M(s)$ t.c. $s[1] \models \phi$
- $s \models \mathbf{E}(\phi\ \mathbf{U}\ \psi)$ if $\exists \sigma \in P_M(s)$ s.t.
  $\exists j \geq 0.\ \sigma[j] \models \psi \wedge (\forall 0 \leq k < j.\ \sigma[k] \models \phi)$

$P_M(s)$ is the set of infinite paths from s in M

Satisfiability

Fixed $M = \langle S, R, L \rangle$, the relation $M, s \models \varphi$ (M satisfies φ in s) is defined as

- $s \models p$ if $p \in L(s)$
- $s \models \neg\phi$ if $s \not\models \phi$
- $s \models \phi \vee \psi$ if $s \models \phi$ or $s \models \psi$
- $s \models \mathbf{EX}\phi$ if $\exists \sigma \in P_M(s)$ t.c. $s[1] \models \phi$
- $s \models \mathbf{E}(\phi \ \mathbf{U} \ \psi)$ if $\exists \sigma \in P_M(s)$ s.t.
  $$\exists j \geq 0.\ \sigma[j] \models \psi \wedge (\forall 0 \leq k < j.\ \sigma[k] \models \phi)$$
- $s \models \mathbf{A}(\phi \ \mathbf{U} \ \psi)$ if $\forall \sigma \in P_M(s)$
  $$\exists j \geq 0.\ \sigma[j] \models \psi \wedge (\forall 0 \leq k < j.\ \sigma[k] \models \phi)$$

$P_M(s)$ is the set of infinite paths from s in M

Other formulas

- $s \models \mathbf{EF}\varphi$ if $\exists \sigma \in P_M(s)(\exists j \geq 0. \sigma[j] \models \varphi)$

Other formulas

- $s \models \mathbf{EF}\varphi$ if $\exists\sigma \in P_M(s)(\exists j \geq 0.\sigma[j] \models \varphi)$
- $s \models \mathbf{EG}\varphi$ if $\exists\sigma \in P_M(s)(\forall j \geq 0.\sigma[j] \models \varphi)$

Other formulas

- $s \models \mathbf{EF}\varphi$ if $\exists \sigma \in P_M(s)(\exists j \geq 0.\sigma[j] \models \varphi)$
- $s \models \mathbf{EG}\varphi$ if $\exists \sigma \in P_M(s)(\forall j \geq 0.\sigma[j] \models \varphi)$
- $s \models \mathbf{AF}\varphi$ if $\forall \sigma \in P_M(s)(\exists j \geq 0.\sigma[j] \models \varphi)$

Other formulas

- $s \models \mathbf{EF}\varphi$ if $\exists\sigma \in P_M(s)(\exists j \geq 0.\sigma[j] \models \varphi)$
- $s \models \mathbf{EG}\varphi$ if $\exists\sigma \in P_M(s)(\forall j \geq 0.\sigma[j] \models \varphi)$
- $s \models \mathbf{AF}\varphi$ if $\forall\sigma \in P_M(s)(\exists j \geq 0.\sigma[j] \models \varphi)$
- $s \models \mathbf{AG}\varphi$ if $\forall\sigma \in P_M(s)(\forall j \geq 0.\sigma[j] \models \varphi)$

CTL Model Checking Algorithms

CTL Model Checking

- The CTL model checking algorithm computes all states of the model that satisfy the property
- The problem can be reduced to that of solving fixpoint equations over monotone functions

Denotation of CTL formulas

Fixed $M = \langle S, R, L\rangle$ and a CTL formula φ,

- We define

$$[\![\varphi]\!] = \{s \mid M, s \models \varphi\}$$

Denotation of CTL formulas

Fixed $M = \langle S, R, L \rangle$ and a CTL formula φ,

- We define

$$[\![\varphi]\!] = \{s \mid M, s \models \varphi\}$$

- Furthermore,

$$\varphi \sqsubseteq \psi \quad \text{iff} \quad [\![\varphi]\!] \subseteq [\![\psi]\!]$$

Denotation of CTL formulas

Fixed $M = \langle S, R, L\rangle$ and a CTL formula φ,

- We define
$$[\![\varphi]\!] = \{s \mid M, s \models \varphi\}$$
- Furthermore,
$$\varphi \sqsubseteq \psi \quad \text{iff} \quad [\![\varphi]\!] \subseteq [\![\psi]\!]$$
- The set of CTL formulas equipped with $\sqsubseteq$ form a complete lattice
  - Least upper bound $= \vee$, indeed $[\![\varphi \vee \psi]\!] = [\![\varphi]\!] \cup [\![\psi]\!]$
  - Greatest lower bound $= \wedge$, indeed, $[\![\varphi \wedge \psi]\!] = [\![\varphi]\!] \cap [\![\psi]\!]$
  - Bottom $[\![\mathit{false}]\!] = \emptyset$
  - Top $[\![\mathit{true}]\!] = S$

Denotations as Fixpoints

- $\mathbf{E}(\varphi \ \mathbf{U} \ \psi) \equiv \psi \vee (\varphi \wedge (\mathbf{EX}(\mathbf{E}[\varphi \ \mathbf{U} \ \psi])))$

Denotations as Fixpoints

- $\mathbf{E}(\varphi\ \mathbf{U}\ \psi) \equiv \psi \vee (\varphi \wedge (\mathbf{EX}(\mathbf{E}[\varphi\ \mathbf{U}\ \psi])))$
- $F(Z) = [\![\psi]\!] \cup ([\![\varphi]\!] \cap Pre_{\exists}(Z))$

Denotations as Fixpoints

- $\mathbf{E}(\varphi\ \mathbf{U}\ \psi) \equiv \psi \vee (\varphi \wedge (\mathbf{EX}(\mathbf{E}[\varphi\ \mathbf{U}\ \psi])))$
- $F(Z) = [\![\psi]\!] \cup ([\![\varphi]\!] \cap \mathit{Pre}_{\exists}(Z))$
- $\mathit{Pre}_{\exists}(Z) = \{s \in S \mid \langle s, s' \rangle \in R, s' \in Z\}$ (predecessors of Z)

Denotations as Fixpoints

- $\mathbf{E}(\varphi \ \mathbf{U} \ \psi) \equiv \psi \vee (\varphi \wedge (\mathbf{EX}(\mathbf{E}[\varphi \ \mathbf{U} \ \psi])))$
- $F(Z) = [\![\psi]\!] \cup ([\![\varphi]\!] \cap Pre_{\exists}(Z))$
- $Pre_{\exists}(Z) = \{s \in S \mid \langle s, s' \rangle \in R, s' \in Z\}$ (predecessors of Z)
- Solve the equation $Z = F(Z)$ where F is monotone

Denotations as Fixpoints

- $\mathbf{E}(\varphi\ \mathbf{U}\ \psi) \equiv \psi \vee (\varphi \wedge (\mathbf{EX}(\mathbf{E}[\varphi\ \mathbf{U}\ \psi])))$
- $F(Z) = [\![\psi]\!] \cup ([\![\varphi]\!] \cap Pre_{\exists}(Z))$
- $Pre_{\exists}(Z) = \{s \in S \mid \langle s, s' \rangle \in R, s' \in Z\}$ (predecessors of Z)
- Solve the equation $Z = F(Z)$ where F is monotone
- $[\![\mathbf{E}(\varphi\ \mathbf{U}\ \psi)]\!]$ is the **least fixpoint** of F (i.e. the smallest set of states A such that $A = F(A)$)

Other formulas: Reachability

- $\mathbf{EF}(\psi) \equiv \psi \vee \mathbf{EX}(\mathbf{EF}\psi)$

Other formulas: Reachability

- $\mathbf{EF}(\psi) \equiv \psi \vee \mathbf{EX}(\mathbf{EF}\psi)$
- $F_1(Z) = [\![\psi]\!] \cup \mathit{Pre}_{\exists}(Z)$

Other formulas: Reachability

- $\mathbf{EF}(\psi) \equiv \psi \vee \mathbf{EX}(\mathbf{EF}\psi)$
- $F_1(Z) = [\![\psi]\!] \cup \mathit{Pre}_{\exists}(Z)$
- Solve the equation $Z = F_1(Z)$ where F_1 is monotone

Other formulas: Reachability

- $\mathbf{EF}(\psi) \equiv \psi \vee \mathbf{EX}(\mathbf{EF}\psi)$
- $F_1(Z) = [\![\psi]\!] \cup Pre_{\exists}(Z)$
- Solve the equation $Z = F_1(Z)$ where F_1 is monotone
- $[\![\mathbf{EF}\psi]\!]$ is the **least fixpoint** of F_1 (i.e. the smallest set of states A such that $A = F_1(A)$

Other formulas

- $[\![\mathbf{EG}\varphi]\!]$ is the greatest fixpoint of $F_2(Z) = [\![\varphi]\!] \cap Pre_\exists(Z)$

Other formulas

- $[\![\mathbf{EG}\varphi]\!]$ is the greatest fixpoint of $F_2(Z) = [\![\varphi]\!] \cap Pre_{\exists}(Z)$
- $[\![\mathbf{AG}\varphi]\!]$ is the greatest fixpoint of $F_3(Z) = [\![\varphi]\!] \cap Pre_{\forall}(Z)$

Other formulas

- $[\![\mathbf{EG}\varphi]\!]$ is the greatest fixpoint of $F_2(Z) = [\![\varphi]\!] \cap Pre_{\exists}(Z)$
- $[\![\mathbf{AG}\varphi]\!]$ is the greatest fixpoint of $F_3(Z) = [\![\varphi]\!] \cap Pre_{\forall}(Z)$

- $Pre_{\forall}(Z) = \{s \in S \mid \forall s' s.t. R(s, s'), s' \in Z\}$

How to compute fixpoints

- Functions $F, F_1, F_2, \ldots$ are monotone w.r.t. inclusion of denotations

How to compute fixpoints

- Functions $F, F_1, F_2, \ldots$ are monotone w.r.t. inclusion of denotations
- We can exploit results from Fixpoint Theory

How to compute fixpoints

- Functions $F, F_1, F_2, \ldots$ are monotone w.r.t. inclusion of denotations
- We can exploit results from Fixpoint Theory
- The least fixpoint of (monotone) F on a finite lattice is the *least upper bound* (lub) (i.e. the union) of the sequence

$$\emptyset \subseteq F(\emptyset) \subseteq F(F(\emptyset)) \ \ldots$$

How to compute fixpoints

- Functions $F, F_1, F_2, \ldots$ are monotone w.r.t. inclusion of denotations
- We can exploit results from Fixpoint Theory
- The least fixpoint of (monotone) F on a finite lattice is the *least upper bound* (lub) (i.e. the union) of the sequence

$$\emptyset \subseteq F(\emptyset) \subseteq F(F(\emptyset)) \ \ldots$$

- The greatest fixpoint is the *greatest lower bound* (glb) (intersection) of the sequence

$$S \supseteq F(S) \supseteq F(F(S)) \ \ldots$$

where S is the set of all states of the model

CTL Model Checking

- Fixed a model M, a state s, and a formula φ, decide if $M, s \models \varphi$

CTL Model Checking

- Fixed a model M, a state s, and a formula φ, decide if $M, s \models \varphi$
- Emerson-Clarke defined the following algorithm: every state s in M is labeled with the set of subformulas of φ that are true in s

CTL Model Checking

- Fixed a model M, a state s, and a formula φ, decide if $M, s \models \varphi$
- Emerson-Clarke defined the following algorithm: every state s in M is labeled with the set of subformulas of φ that are true in s
- The labeling is built inductively starting from the subformulas of minimal size (atomic formulas)

CTL Model Checking

- Fixed $M = \langle S, R, L \rangle$, let AP be the set of atomic formulas
- The algorithm is based on the function *Sat* that computes the set of states that satisfies a formula φ

---

function Sat(φ : CTL formula) : set of states
begin
  if $\varphi =$ true then return S
  if $\varphi =$ false then return $\emptyset$
  if $\varphi \in$ AP then return $\{ s \mid \varphi \in L(s) \}$
  if $\varphi = \neg\psi$ then return $S \setminus \mathrm{Sat}(\psi)$
  if $\varphi = \varphi_1 \vee \varphi_2$ then return $\mathrm{Sat}(\varphi_1) \cup \mathrm{Sat}(\varphi_2)$
  if $\varphi = \mathbf{EX}\varphi_1$ then return $\mathrm{Pre}_\exists(\mathrm{Sat}(\varphi_1))$
  if $\varphi = \mathbf{E}(\varphi_1 \ \mathbf{U}\ \varphi_2)$ then return $\mathrm{Sat}_{\mathrm{EU}}(\varphi_1, \varphi_2)$
  if $\varphi = \mathbf{A}(\varphi_1 \ \mathbf{U}\ \varphi_2)$ then return $\mathrm{Sat}_{\mathrm{AU}}(\varphi_1, \varphi_2)$
end

Procedure for **EU**

```
function Sat_EU(φ1, φ2 : CTL formula) : set of states
var Q, Q' : set of states
begin
Q := Sat(φ2)
Q' := ∅
while Q ≠ Q' do
    Q' := Q;
    Q := Q ∪ (Sat(φ1) ∩ Pre_∃(Q));
endw;
return Q;
end
```

Procedure for **AU**

```
function Sat_AU(φ1, φ2 : CTL formula) : set of states
var Q, Q' : set of states
begin
Q := Sat(φ2)
Q' := ∅
while Q ≠ Q' do
    Q' := Q;
    Q := Q ∪ (Sat(φ1) ∩ Pre_∀(Q));
endw;
return Q;
end
```

Complexity

- Model checking a CTL formula φ against a model M has time complexity $O(|M| \times |\varphi|)$
- **Termination**
  In principle it is not a problem for finite-state systems
  In practice: state-explosion problem

Symbolic Model Checking

- **Symbolic Representation**
  State = assignment to Boolean variables
  Transition relation= Boolean formula
  Predecessor relation $Pre_{\exists}$ = Existentially quantified formula

$$Pre_{\exists}(F(x)) = \exists y. T(x, y) \wedge F([y/x])$$

  where $T(x, y)$ is the transition relation
- **Symbolic Model Checking Algorithm**
  Fixpoint computation using Boolean formulas as symbolic representation of *finite sets* of states
- CTL model checkers like SMV, nuSMV, Mucke are based on OBDDs