Branching Time: CTL

Natasha Alechina   Brian Logan

Utrecht University

n.a.alechina@uu.nl   b.s.logan@uu.nl

Branching Time: CTL and CTL*

- **CTL**: **Computation Tree Logic**
- Reasoning about all possible computations of a system

- Path quantifiers: *A* (for all paths), *E* (there is a path)
- Temporal operators: *X* (next), *F* (sometime), *G* (always), *U* (until)

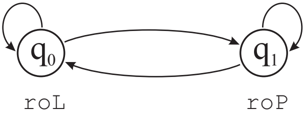
- "Vanilla" CTL: every temporal operator must be immediately preceded by exactly one path quantifier
- CTL*: no syntactic restrictions; two kinds of formulas: state formulas vs. path formulas
- model checking CTL is much easier

Branching Time: CTL and CTL*

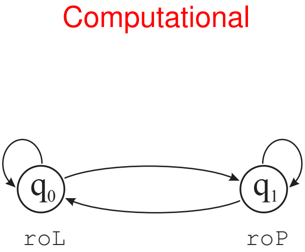
- Models: transition systems; include: states (time points, situations), transitions (changes)
- Paths: full sequences of states that can be produced by following transitions in the transition system

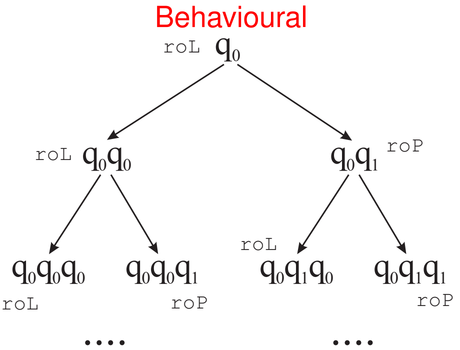
Computational vs. Behavioral Structures

Computational



Computational vs. Behavioral Structures





Computational vs. Behavioural Structures

In CTL/CTL*, models are defined as computational structures!

Semantics of CTL*

Definition (Semantics of CTL*: state formulae)

$M, q \models E\gamma$ iff there is a path λ, starting from q, such that $M, \lambda \models \gamma$;
$M, q \models A\gamma$ iff for all paths λ, starting from q, we have $M, \lambda \models \gamma$.

Definition (Semantics of CTL*: path formulae)

Semantics of CTL*

Definition (Semantics of CTL*: state formulae)

$M, q \models E\gamma$ iff there is a path λ, starting from q, such that $M, \lambda \models \gamma$;
$M, q \models A\gamma$ iff for all paths λ, starting from q, we have $M, \lambda \models \gamma$.

Definition (Semantics of CTL*: path formulae)

Like in LTL!

Semantics of CTL*

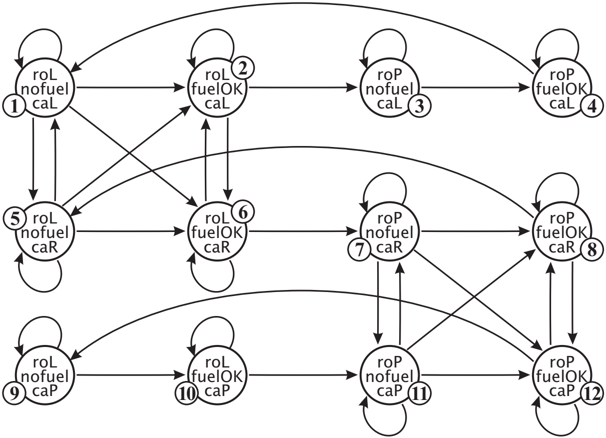
Definition (Semantics of CTL*: state formulae)

$M, q \models E\gamma$	iff there is a path λ, starting from q, such that $M, \lambda \models \gamma$;
$M, q \models A\gamma$	iff for all paths λ, starting from q, we have $M, \lambda \models \gamma$.

Definition (Semantics of CTL*: path formulae)

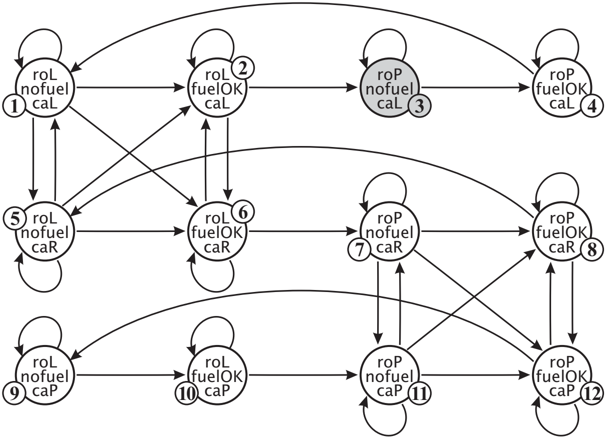
$M, \lambda \models \varphi$	iff $M, \lambda[0] \models \varphi$, for a state formula φ;
$M, \lambda \models X\varphi$	iff $M, \lambda[1...\infty] \models \varphi$;
$M, \lambda \models F\varphi$	iff $M, \lambda[i..\infty] \models \varphi$ for some $i \geq 0$;
$M, \lambda \models G\varphi$	iff $M, \lambda[i..\infty] \models \varphi$ for all $i \geq 0$;
$M, \lambda \models \varphi U \psi$	iff $M, \lambda[i...\infty] \models \psi$ for some $i \geq 0$, and $M, \lambda[j...\infty] \models \varphi$ for all $0 \leq j < i$.

Example: Rocket and Cargo



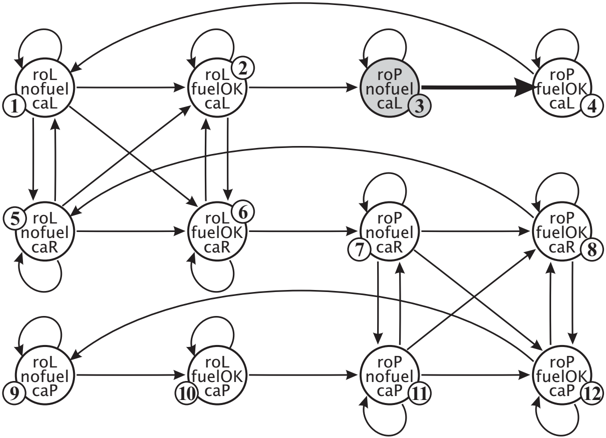
EF caP

Example: Rocket and Cargo



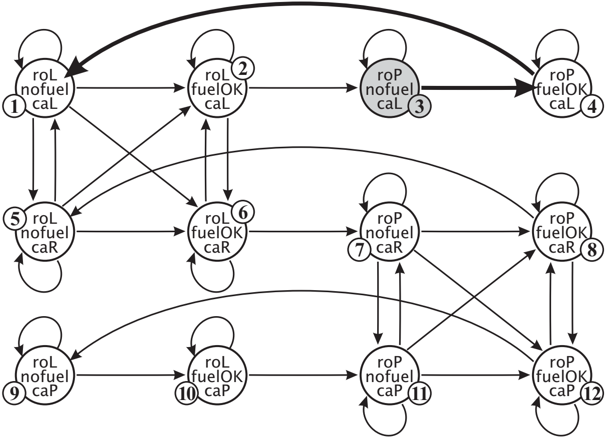
$EF\,$caP

Example: Rocket and Cargo



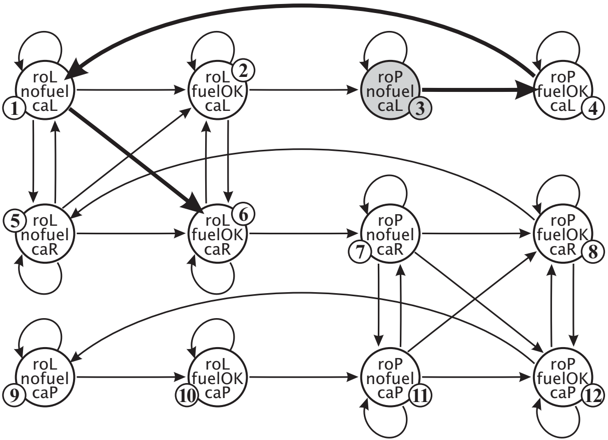
$EF\,$caP

Example: Rocket and Cargo



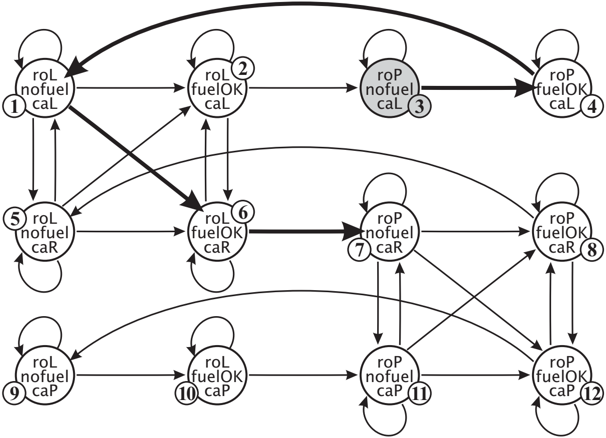
EF caP

Example: Rocket and Cargo



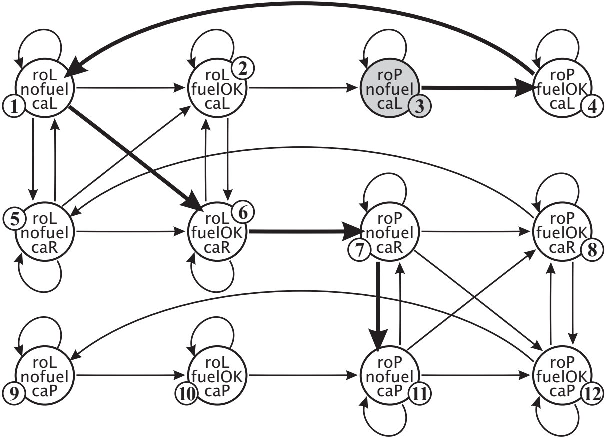
$EF\,\mathrm{caP}$

Example: Rocket and Cargo



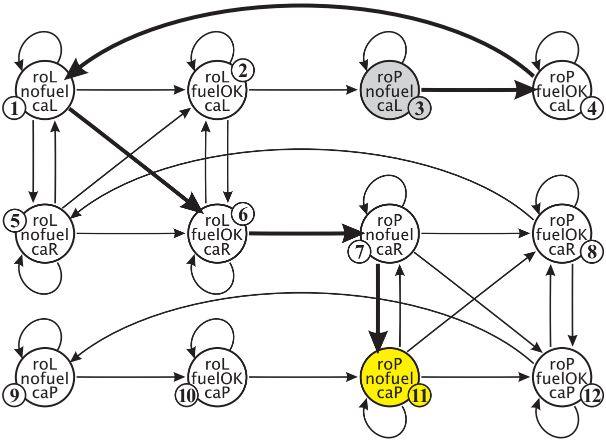
$EF\,$caP

Example: Rocket and Cargo



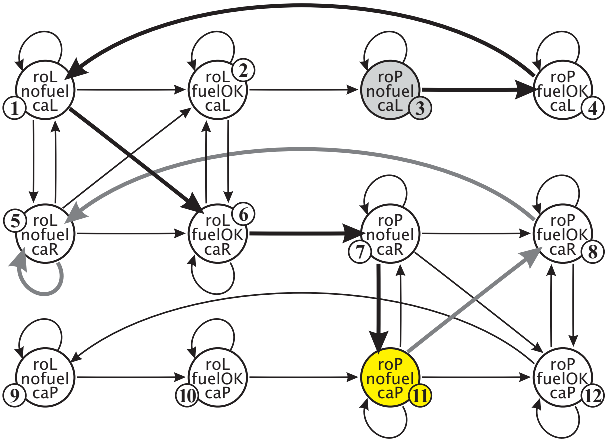
$EF\,$caP

Example: Rocket and Cargo



$EF\,\mathrm{caP}$

Example: Rocket and Cargo



EF caP

Alternative Semantics for “Vanilla” CTL

The semantics of **“vanilla” CTL** can be given entirely with respect to states in the model:

$M, q \models EX\varphi$ iff there is a path λ starting from q, such that $M, \lambda[1] \models \varphi$;

Alternative Semantics for “Vanilla” CTL

The semantics of **“vanilla” CTL** can be given entirely with respect to states in the model:

$M, q \models EX\varphi$ iff there is a path λ starting from q, such that $M, \lambda[1] \models \varphi$;

Alternative Semantics for "Vanilla" CTL

The semantics of **"vanilla" CTL** can be given entirely with respect to states in the model:

$M, q \models EX\varphi$	iff there is a path λ starting from q, such that $M, \lambda[1] \models \varphi$;
$M, q \models EF\varphi$	iff there is a path λ starting from q, such that $M, \lambda[i] \models \varphi$ for some $i \geq 0$;

Alternative Semantics for "Vanilla" CTL

The semantics of **"vanilla" CTL** can be given entirely with respect to states in the model:

$M, q \models EX\varphi$	iff there is a path λ starting from q, such that $M, \lambda[1] \models \varphi$;
$M, q \models EF\varphi$	iff there is a path λ starting from q, such that $M, \lambda[i] \models \varphi$ for some $i \geq 0$;
$M, q \models EG\varphi$	iff there is a path λ starting from q, such that $M, \lambda[i] \models \varphi$ for all $i \geq 0$;

Alternative Semantics for "Vanilla" CTL

The semantics of **"vanilla" CTL** can be given entirely with respect to states in the model:

$M, q \models EX\varphi$	iff there is a path λ starting from q, such that $M, \lambda[1] \models \varphi$;
$M, q \models EF\varphi$	iff there is a path λ starting from q, such that $M, \lambda[i] \models \varphi$ for some $i \geq 0$;
$M, q \models EG\varphi$	iff there is a path λ starting from q, such that $M, \lambda[i] \models \varphi$ for all $i \geq 0$;
$M, q \models E\varphi U \psi$	iff there is a path λ starting from q, such that $M, \lambda[i] \models \psi$ for some $i \geq 0$, and $M, \lambda[j] \models \varphi$ for all $0 \leq j < i$.

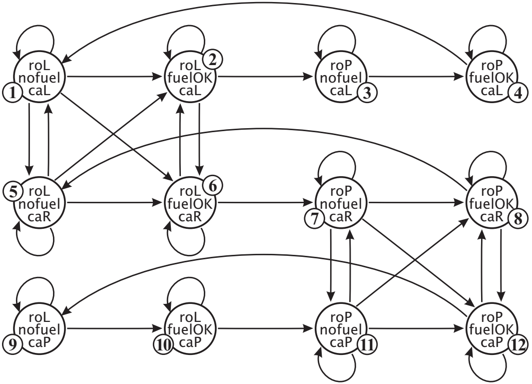
Alternative Semantics for "Vanilla" CTL

The semantics of **"vanilla" CTL** can be given entirely with respect to states in the model:

$M, q \models EX\varphi$	iff there is a path λ starting from q, such that $M, \lambda[1] \models \varphi$;
$M, q \models EF\varphi$	iff there is a path λ starting from q, such that $M, \lambda[i] \models \varphi$ for some $i \geq 0$;
$M, q \models EG\varphi$	iff there is a path λ starting from q, such that $M, \lambda[i] \models \varphi$ for all $i \geq 0$;
$M, q \models E\varphi U \psi$	iff there is a path λ starting from q, such that $M, \lambda[i] \models \psi$ for some $i \geq 0$, and $M, \lambda[j] \models \varphi$ for all $0 \leq j < i$.

...and analogously for AX, AF, AG, AU.

Rocket and Cargo: More Properties



$\text{roL} \rightarrow EF\,\text{roP}$

$AG\,(\text{roL} \vee \text{roP})$

$\text{roL} \rightarrow AX(\text{roP} \rightarrow \text{nofuel})$

Motivating Example: Rescue Robots

- Everybody is safe

  $\bigwedge_{i \in People} \mathsf{safe_i}$

- Everybody will eventually be safe

  $\bigwedge_{i \in People} AF\, \mathsf{safe_i}$

  Another interpretation: $AF\,(\bigwedge_{i \in People} \mathsf{safe_i})$

- Everybody will always be safe, from some moment on

  $\bigwedge_{i \in People} AFG\, \mathsf{safe_i}$

  Equivalently: $AFG\,(\bigwedge_{i \in People} \mathsf{safe_i})$

- Everybody may eventually be safe, if everything goes fine

  $\bigwedge_{i \in People} EF\, \mathsf{safe_i}$

  Another interpretation: $EF\,(\bigwedge_{i \in People} \mathsf{safe_i})$

Motivating Example: Rescue Robots

- Whenever person *i* gets in trouble, she will eventually be rescued

  $AG\,(\neg \mathrm{safe}_i \rightarrow AF\,\mathrm{safe}_i)$

- If person *i* gets outside the building then she will never be in danger anymore

  $AG\,(\mathrm{outside}_i \rightarrow AG\,\mathrm{safe}_i)$

- Person *i* may be rescued without any robot ever entering the building, but guaranteed rescue requires some robots to enter

  $E(\bigwedge_{j \in Robots} \mathrm{outside}_j)\,U\,\mathrm{safe}_i \ \wedge\ \neg A(\bigwedge_{j \in Robots} \mathrm{outside}_j)\,U\,\mathrm{safe}_i$

Fixpoint Equivalences in CTL

Theorem (Fixpoint characterization of branching-time operators)

*The following formulae are **valid** in CTL:*

- $EF\varphi \quad \leftrightarrow \quad \varphi \vee EX\, EF\varphi$
- $EG\,\varphi \quad \leftrightarrow \quad \varphi \wedge EX\, EA\varphi$
- $E\varphi_1\, U\, \varphi_2 \quad \leftrightarrow \quad \varphi_2 \vee (\varphi_1 \wedge EX\, E\varphi_1\, U\, \varphi_2)$.

- $AF\varphi \quad \leftrightarrow \quad \varphi \vee AX\, AF\varphi$
- $AG\,\varphi \quad \leftrightarrow \quad \varphi \wedge AX\, AG\,\varphi$
- $A\varphi_1\, U\, \varphi_2 \quad \leftrightarrow \quad \varphi_2 \vee (\varphi_1 \wedge AX\, A\varphi_1\, U\, \varphi_2)$.

Fixpoint Equivalences in CTL

What is the importance of fixpoint equivalences?

- they say that paths satisfying CTL specifications can be constructed incrementally, step by step
- moreover, solutions to the verification problem can be obtained iteratively
- ...which will be used in most model checking algorithms (see the next lecture)