Modal Logics for Multi-Agent Systems

Valentin Goranko[1] and Wojtek Jamroga[2]

[1] University of the Witwatersrand, Johannesburg, South Africa

[2] Clausthal University of Technology, Germany

Section 3. Logics of Action and Time

Logics of Action and Time
3.1 Dynamic Logic
3.2 Temporal Logic
3.3 Linear Time Logic
3.4 Computation Tree Logic

Up to now:

- Several operators $\mathbf{K}_i$, each defines an epistemic relation on worlds.

Up to now:

- Several operators $\mathbf{K}_i$, each defines an epistemic relation on worlds.
- Description of static systems: no possibility of change

Up to now:

- Several operators $\mathbf{K}_i$, each defines an epistemic relation on worlds.
- Description of static systems: no possibility of change

But:

- MAS are dynamic!

3.1 Dynamic Logic

1^{st} idea: Consider actions or programs α. Each such α defines a transition (accessibility relation) from worlds into worlds.

1st idea: Consider actions or programs α. Each such α defines a transition (accessibility relation) from worlds into worlds.

2nd idea: We need statements about the outcome of actions:

- $[\alpha]\varphi$: "after every execution of α, φ holds,
- $\langle\alpha\rangle\varphi$: "after some executions of α, φ holds.

1st idea: Consider actions or programs α. Each such α defines a transition (accessibility relation) from worlds into worlds.

2nd idea: We need statements about the outcome of actions:

- $[\alpha]\varphi$: "after every execution of α, φ holds,
- $\langle\alpha\rangle\varphi$: "after some executions of α, φ holds.

As usual, $\langle\alpha\rangle\varphi \equiv \neg[\alpha]\neg\varphi$.

3rd idea: Programs/actions can be combined (sequentially, nondeterministically, iteratively), e.g.:

$$[\alpha; \beta]\varphi$$

would mean "after every execution of α and then β, formula φ holds".

Definition 3.1 (Labelled Transition System)

A labelled transition system is a pair

$$\langle St, \{\xrightarrow{\alpha} : \ \alpha \in \mathbf{L}\}\rangle$$

where St is a non-empty set of states and $\mathbf{L}$ is a non-empty set of labels and for each $\alpha \in \mathbf{L}$:
$\xrightarrow{\alpha} \subseteq St \times St$.

Definition 3.1 (Labelled Transition System)

A labelled transition system is a pair

$$\langle St, \{\xrightarrow{\alpha}: \ \alpha \in \mathbf{L}\}\rangle$$

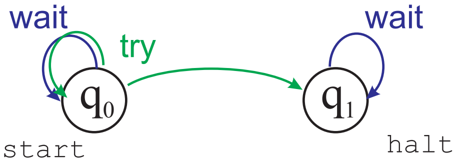
where St is a non-empty set of states and $\mathbf{L}$ is a non-empty set of labels and for each $\alpha \in \mathbf{L}$: $\xrightarrow{\alpha} \subseteq St \times St$.

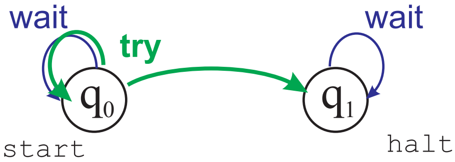
Definition 3.2 (Dynamic Logic: Models)

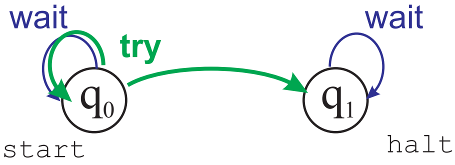
A model of propositional dynamic logic is given by a labelled transition systems and an evaluation of propositions.

Definition 3.3 (Semantics of DL)

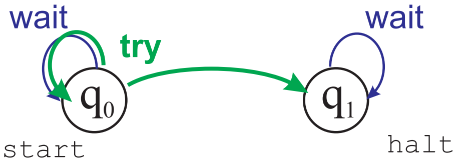
$\mathcal{M}, s \models [\alpha]\varphi$ iff for every t such that $s \xrightarrow{\alpha} t$, we have $\mathcal{M}, t \models \varphi$.


wait
try
wait
q0
q1
start
halt

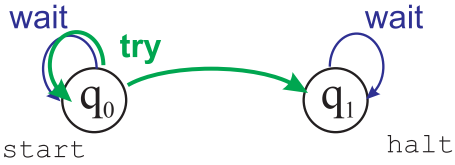

wait
try
wait
q_0
q_1
start
halt



$$\text{start} \rightarrow \langle try \rangle \text{halt}$$



$$\text{start} \rightarrow \langle try \rangle \text{halt}$$
$$\text{start} \rightarrow \neg[try]\text{halt}$$



$$\text{start} \rightarrow \langle \mathit{try} \rangle \text{halt}$$
$$\text{start} \rightarrow \neg[\mathit{try}]\text{halt}$$
$$\text{start} \rightarrow \langle \mathit{try} \rangle [\mathit{wait}]\text{halt}$$

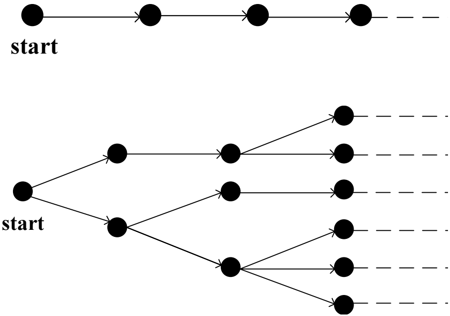
3.2 Temporal Logic

Ideas:

- The accessibility relation can be seen as representing time.
- time: linear vs. branching

Ideas:

- The accessibility relation can be seen as representing time.
- time: linear vs. branching



Typical temporal operators

Typical temporal operators

$\mathcal{X}\varphi$	φ is true in the next moment in time
$\mathcal{G}\varphi$	φ is true in all future moments
$\mathcal{F}\varphi$	φ is true in some future moment
$\varphi\mathcal{U}\psi$	φ is true until the moment when ψ becomes true

Typical temporal operators

$\mathcal{X}\varphi$	φ is true in the next moment in time
$\mathcal{G}\varphi$	φ is true in all future moments
$\mathcal{F}\varphi$	φ is true in some future moment
$\varphi\mathcal{U}\psi$	φ is true until the moment when ψ becomes true

$$\mathcal{G}((\neg\text{passport} \vee \neg\text{ticket}) \rightarrow \mathcal{X}\neg\text{board\_flight})$$

$$\text{send}(\text{msg}, \text{rcvr}) \rightarrow \mathcal{F}\text{receive}(\text{msg}, \text{rcvr})$$

Temporal logic was originally developed in order to represent tense in natural language.

Temporal logic was originally developed in order to represent tense in natural language.

Within Computer Science, it has achieved a significant role in the formal specification and verification of concurrent and distributed systems.

Temporal logic was originally developed in order to represent tense in natural language.

Within Computer Science, it has achieved a significant role in the formal specification and verification of concurrent and distributed systems.

Much of this popularity has been achieved as a number of useful concepts can be formally, and concisely, specified using temporal logics, e.g.

Temporal logic was originally developed in order to represent tense in natural language.

> Within Computer Science, it has achieved a significant role in the formal specification and verification of concurrent and distributed systems.

Much of this popularity has been achieved as a number of useful concepts can be formally, and concisely, specified using temporal logics, e.g.

- safety properties

Temporal logic was originally developed in order to represent tense in natural language.

Within Computer Science, it has achieved a significant role in the formal specification and verification of concurrent and distributed systems.

Much of this popularity has been achieved as a number of useful concepts can be formally, and concisely, specified using temporal logics, e.g.

- safety properties
- liveness properties

Temporal logic was originally developed in order to represent tense in natural language.

> Within Computer Science, it has achieved a significant role in the formal specification and verification of concurrent and distributed systems.

Much of this popularity has been achieved as a number of useful concepts can be formally, and concisely, specified using temporal logics, e.g.

- safety properties
- liveness properties
- fairness properties

Safety:

"something bad will not happen"

"something good will always hold"

Safety:

"something bad will not happen"
"something good will always hold"

Typical examples:

Safety:

"something bad will not happen"
"something good will always hold"

Typical examples:

$\mathcal{G}\neg$bankrupt

Safety:

"something bad will not happen"
"something good will always hold"

Typical examples:

$\mathcal{G}\neg\text{bankrupt}$
$\mathcal{G}(\text{fuelOK} \vee \mathcal{X}\text{fuelOK})$

Safety:

"something bad will not happen"
"something good will always hold"

Typical examples:

$\mathcal{G}\neg$bankrupt
$\mathcal{G}$(fuelOK $\vee$ $\mathcal{X}$fuelOK)
and so on ...

Safety:

"something bad will not happen"
"something good will always hold"

Typical examples:

$\mathcal{G}\neg\text{bankrupt}$
$\mathcal{G}(\text{fuelOK} \vee \mathcal{X}\text{fuelOK})$
and so on . . .

Usually: $\mathcal{G}\neg\ldots.$

Liveness:

"something good will happen"

Liveness:

"something good will happen"

Typical examples:

Liveness:

"something good will happen"

Typical examples:

$\mathcal{F}$rich

Liveness:

"something good will happen"

Typical examples:

$\mathcal{F}\mathsf{rich}$

$\mathsf{rocketLondon} \rightarrow \mathcal{F}\mathsf{rocketParis}$

Liveness:

"something good will happen"

Typical examples:

$\mathcal{F}$rich

rocketLondon $\rightarrow$ $\mathcal{F}$rocketParis

and so on ...

Liveness:

"something good will happen"

Typical examples:

$\mathcal{F}$rich

rocketLondon $\rightarrow \mathcal{F}$rocketParis

and so on . . .

Usually: $\mathcal{F}$....

Combinations of safety and liveness possible:

Combinations of safety and liveness possible:

$\mathcal{F}\mathcal{G}\mathsf{rocketParis}$

$\mathcal{G}(\mathsf{rocketLondon} \rightarrow \mathcal{F}\mathsf{rocketParis})$

Combinations of safety and liveness possible:

$\mathcal{F}\mathcal{G}\text{rocketParis}$

$\mathcal{G}(\text{rocketLondon} \rightarrow \mathcal{F}\text{rocketParis})$ $\rightsquigarrow$ fairness

Strong fairness

"if something is attempted/requested, then it will be successful/allocated"

Strong fairness

"if something is attempted/requested, then it will be successful/allocated"

Typical examples:

$$\mathcal{G}(\text{attempt} \rightarrow \mathcal{F}\text{success})$$

Strong fairness

"if something is attempted/requested, then it will be successful/allocated"

Typical examples:

$$\mathcal{G}(\text{attempt} \rightarrow \mathcal{F}\text{success})$$

$$\mathcal{G}\mathcal{F}\text{attempt} \rightarrow \mathcal{G}\mathcal{F}\text{success}$$

Fairness:

- Useful when scheduling processes, responding to messages, etc.
- Good for specifying properties of the environment.

3.3 Linear Time Logic

Linear Time: LTL

- LTL: Linear Time Logic
- Reasoning about a particular computation of a system
- Time is linear: just one possible future path is included!

Linear Time: LTL

- LTL: Linear Time Logic
- Reasoning about a particular computation of a system
- Time is linear: just one possible future path is included!
- Models: paths

Definition 3.4 (Models of LTL)

A model of LTL is a sequence of time moments. We call such models paths, and denote them by λ.

Definition 3.4 (Models of LTL)

A model of LTL is a sequence of time moments. We call such models paths, and denote them by λ.

Evaluation of atomic propositions at particular time moments is also needed.

Definition 3.4 (Models of LTL)

A model of LTL is a sequence of time moments. We call such models paths, and denote them by λ.

Evaluation of atomic propositions at particular time moments is also needed.

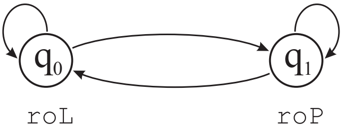
Notation:

- $\lambda[i]$: ith time moment
- $\lambda[i \dots j]$: all time moments between i and j
- $\lambda[i \dots \infty]$: all timepoints from i on

Important: computational vs. behavioral structures

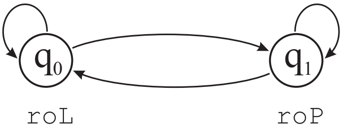
Important: computational vs. behavioral structures

Computational

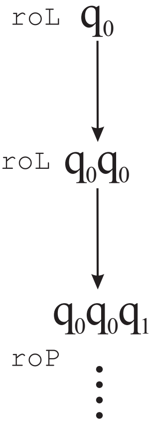


Important: computational vs. behavioral structures

Computational



Behavioral



LTL models are defined as behavioral structures!

Definition 3.5 (Semantics of LTL)

$\lambda \models p$ iff p is true at moment $\lambda[0]$;

Definition 3.5 (Semantics of LTL)

$\lambda \models p$ iff p is true at moment $\lambda[0]$;

$\lambda \models \mathcal{X}\varphi$ iff $\lambda[1..\infty] \models \varphi$;

Definition 3.5 (Semantics of LTL)

$\lambda \models p$ iff p is true at moment $\lambda[0]$;

$\lambda \models \mathcal{X}\varphi$ iff $\lambda[1..\infty] \models \varphi$;

$\lambda \models \mathcal{F}\varphi$ iff $\lambda[i..\infty] \models \varphi$ for some $i \geq 0$;

Definition 3.5 (Semantics of LTL)

$\lambda \models p$	iff p is true at moment $\lambda[0]$;
$\lambda \models \mathcal{X}\varphi$	iff $\lambda[1..\infty] \models \varphi$;
$\lambda \models \mathcal{F}\varphi$	iff $\lambda[i..\infty] \models \varphi$ for some $i \geq 0$;
$\lambda \models \mathcal{G}\varphi$	iff $\lambda[i..\infty] \models \varphi$ for all $i \geq 0$;

Definition 3.5 (Semantics of LTL)

$\lambda \models p$	iff p is true at moment $\lambda[0]$;
$\lambda \models \mathcal{X}\varphi$	iff $\lambda[1..\infty] \models \varphi$;
$\lambda \models \mathcal{F}\varphi$	iff $\lambda[i..\infty] \models \varphi$ for some $i \geq 0$;
$\lambda \models \mathcal{G}\varphi$	iff $\lambda[i..\infty] \models \varphi$ for all $i \geq 0$;
$\lambda \models \varphi\mathcal{U}\psi$	iff $\lambda[i..\infty] \models \psi$ for some $i \geq 0$, and $\lambda[j..\infty] \models \varphi$ for all $0 \leq j \leq i$.

Definition 3.5 (Semantics of LTL)

$\lambda \models p$	iff p is true at moment $\lambda[0]$;
$\lambda \models \mathcal{X}\varphi$	iff $\lambda[1..\infty] \models \varphi$;
$\lambda \models \mathcal{F}\varphi$	iff $\lambda[i..\infty] \models \varphi$ for some $i \geq 0$;
$\lambda \models \mathcal{G}\varphi$	iff $\lambda[i..\infty] \models \varphi$ for all $i \geq 0$;
$\lambda \models \varphi\mathcal{U}\psi$	iff $\lambda[i..\infty] \models \psi$ for some $i \geq 0$, and $\lambda[j..\infty] \models \varphi$ for all $0 \leq j \leq i$.

Note that:

$$\mathcal{G}\varphi \equiv \neg\mathcal{F}\neg\varphi$$

$$\mathcal{F}\varphi \equiv \neg\mathcal{G}\neg\varphi$$

$$\mathcal{F}\varphi \equiv \top\mathcal{U}\varphi$$

3.4 Computation Tree Logic

Branching Time: CTL

- CTL: Computation Tree Logic.
- Reasoning about possible computations of a system
- Time is branching: we want all alternative paths included!

Branching Time: CTL

- CTL: Computation Tree Logic.
- Reasoning about possible computations of a system
- Time is branching: we want all alternative paths included!
- Models: states (time points, situations), transitions (changes)
- Paths: courses of action, computations.

- Path quantifiers: **A** (for all paths), **E** (there is a path);
- Temporal operators: $\mathcal{X}$ (nexttime), $\mathcal{F}$ (sometime), $\mathcal{G}$ (always) and $\mathcal{U}$ (until);

- Path quantifiers: **A** (for all paths), **E** (there is a path);
- Temporal operators: $\mathcal{X}$ (nexttime), $\mathcal{F}$ (sometime), $\mathcal{G}$ (always) and $\mathcal{U}$ (until);

- "Vanilla" CTL: every temporal operator must be immediately preceded by exactly one path quantifier;
- CTL*: no syntactic restrictions;

- Path quantifiers: **A** (for all paths), **E** (there is a path);
- Temporal operators: $\mathcal{X}$ (nexttime), $\mathcal{F}$ (sometime), $\mathcal{G}$ (always) and $\mathcal{U}$ (until);

- "Vanilla" CTL: every temporal operator must be immediately preceded by exactly one path quantifier;
- CTL*: no syntactic restrictions;
- Reasoning in "vanilla" CTL can be automatized.

Definition 3.6 (CTL models: transition systems)

A transition system is a pair

$$\langle St, \longrightarrow \rangle$$

where:

- St is a non-empty set of states,
- $\longrightarrow \subseteq St \times St$ is a transition relation.

Definition 3.6 (CTL models: transition systems)

A transition system is a pair

$$\langle St, \longrightarrow \rangle$$

where:

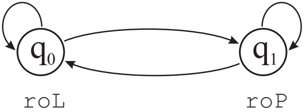
- St is a non-empty set of states,
- $\longrightarrow \subseteq St \times St$ is a transition relation.

Note that, formally, transition relation is just a modal accessibility relation.

Important: computational vs. behavioral structures

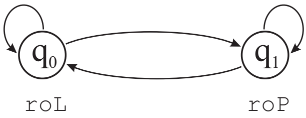
Important: computational vs. behavioral structures

Computational

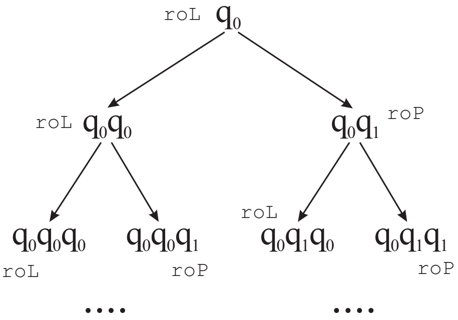


Important: computational vs. behavioral structures

Computational



Behavioral



CTL models are defined as computational structures!

Definition 3.7 (Paths in a model)

A path λ is an infinite sequence of states that can be effected by subsequent transitions.

A path must be full, i.e. either infinite, or ending in a state with no outgoing transition.

Definition 3.7 (Paths in a model)

A path λ is an infinite sequence of states that can be effected by subsequent transitions.

A path must be full, i.e. either infinite, or ending in a state with no outgoing transition.

Usually, we assume that the transition relation is serial (time flows forever).

Definition 3.7 (Paths in a model)

A path λ is an infinite sequence of states that can be effected by subsequent transitions.

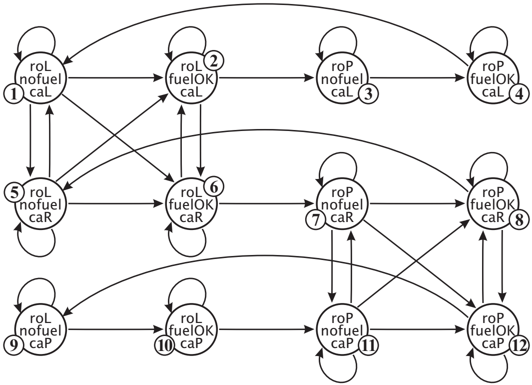
A path must be full, i.e. either infinite, or ending in a state with no outgoing transition.

Usually, we assume that the transition relation is serial (time flows forever).
Then, all paths are infinite.

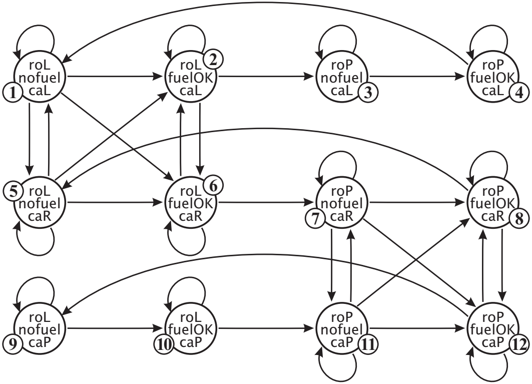
Example: Rocket and Cargo

- A rocket and a cargo,
- The rocket can be moved between London (proposition roL) and Paris (proposition roP),
- The cargo can be in London (caL), Paris (caP), or inside the rocket (caR),
- The rocket can be moved only if it has its fuel tank full (fuelOK),
- When it moves, it consumes fuel, and nofuel holds after each flight.

Example: Rocket and Cargo

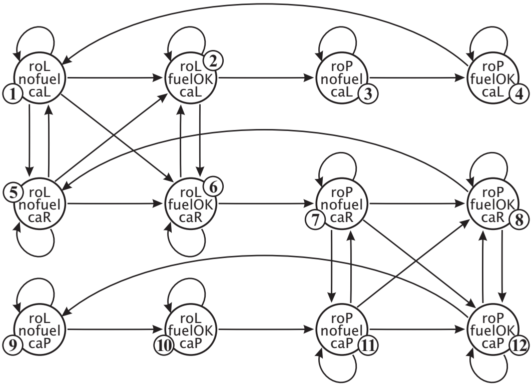


Example: Rocket and Cargo



$$\mathsf{roL} \rightarrow \mathbf{E}\Diamond\mathsf{roP}$$

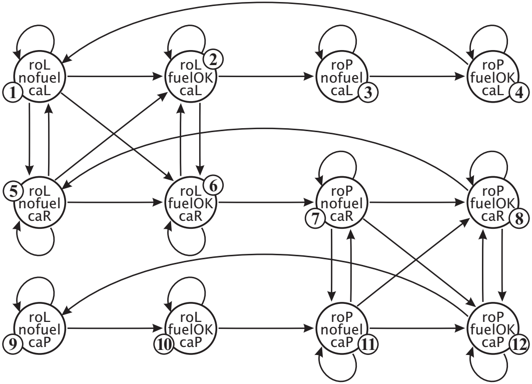
Example: Rocket and Cargo



$\mathsf{roL} \rightarrow \mathbf{E}\Diamond\mathsf{roP}$

$\mathbf{A}\mathcal{G}(\mathsf{roL} \vee \mathsf{roP})$

Example: Rocket and Cargo



$$\mathsf{roL} \rightarrow \mathbf{E}\Diamond \mathsf{roP}$$

$$\mathbf{A}\mathcal{G}(\mathsf{roL} \vee \mathsf{roP})$$

$$\mathsf{roL} \rightarrow \mathbf{A}\mathcal{X}(\mathsf{roP} \rightarrow \mathsf{nofuel})$$

Definition 3.8 (Semantics of CTL*: state formulae)

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

Definition 3.8 (Semantics of CTL*: state formulae)

$M, q \models \mathbf{E}\varphi$ iff there is a path λ, starting from q, such that $M, \lambda \models \varphi$;

$M, q \models \mathbf{A}\varphi$ iff for all paths λ, starting from q, we have $M, \lambda \models \varphi$.

Definition 3.9 (Semantics of CTL*: path formulae)

Exactly like for LTL!

Definition 3.8 (Semantics of CTL*: state formulae)

$M, q \models \mathbf{E}\varphi$ iff there is a path λ, starting from q, such that $M, \lambda \models \varphi$;

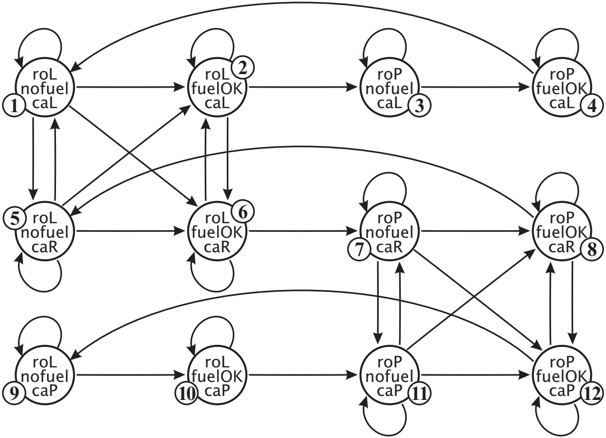
$M, q \models \mathbf{A}\varphi$ iff for all paths λ, starting from q, we have $M, \lambda \models \varphi$.

Definition 3.9 (Semantics of CTL*: path formulae)

$M, \lambda \models \mathcal{X}\varphi$ iff $M, \lambda[1...\infty] \models \varphi$;

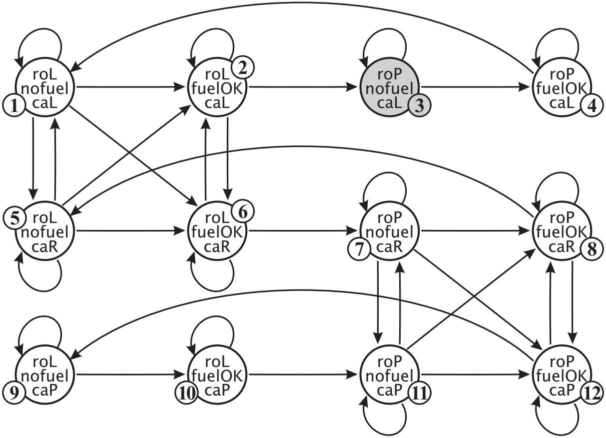
$M, \lambda \models \varphi\,\mathcal{U}\psi$ iff $M, \lambda[i...\infty] \models \psi$ for some $i \geq 0$, and $M, \lambda[j...\infty] \models \varphi$ for all $0 \leq j \leq i$.

Example: Rocket and Cargo



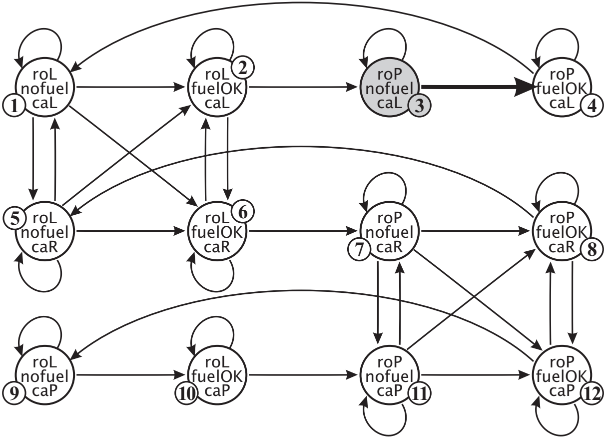
$\mathsf{E}\Diamond \mathsf{caP}$

Example: Rocket and Cargo



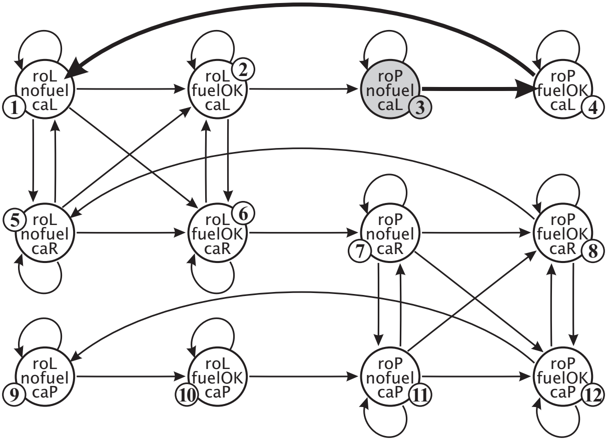
$\mathsf{E}\Diamond \mathrm{caP}$

Example: Rocket and Cargo



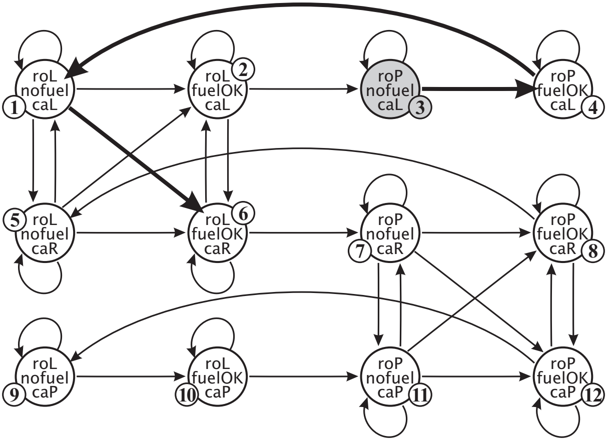
$\mathsf{E}\Diamond \mathsf{caP}$

Example: Rocket and Cargo



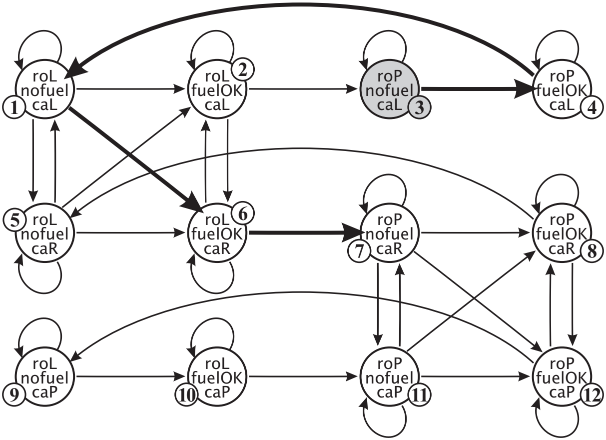
$\mathsf{E}\Diamond \mathrm{caP}$

Example: Rocket and Cargo



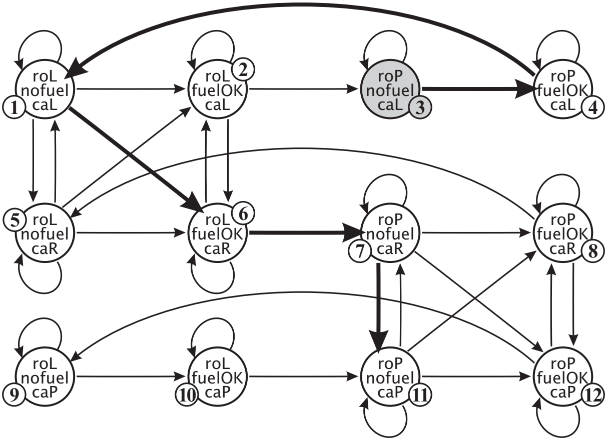
$\mathsf{E}\Diamond \mathsf{caP}$

Example: Rocket and Cargo



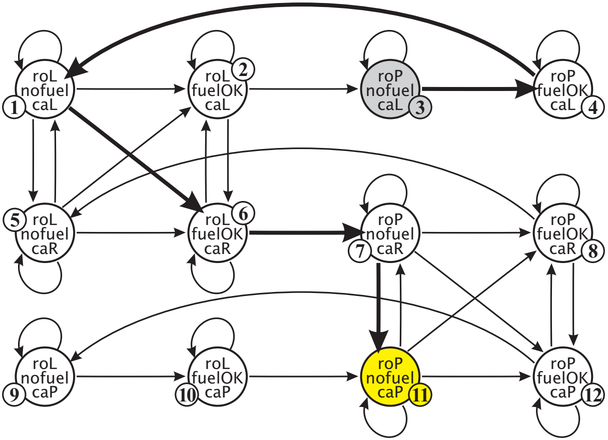
$\mathsf{E}\Diamond \mathsf{caP}$

Example: Rocket and Cargo



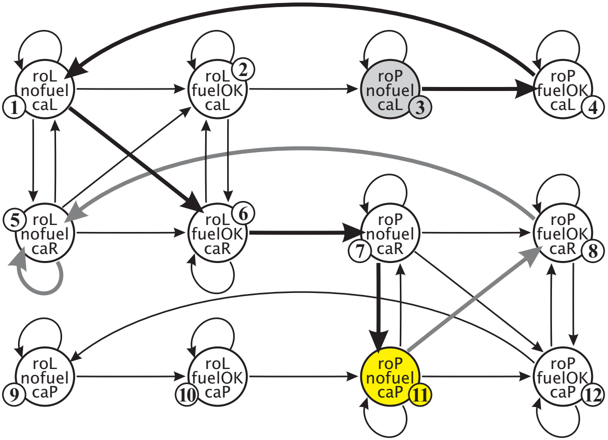
E◊caP

Example: Rocket and Cargo



$\mathsf{E}\Diamond \mathsf{caP}$

Example: Rocket and Cargo



$\mathsf{E}\Diamond \mathsf{caP}$

Exercise:

How to express that there is no possibility of a deadlock?

Practical Importance of Temporal and Dynamic Logics:

Automatic verification in principle possible (model checking).

Practical Importance of Temporal and Dynamic Logics:

Automatic verification in principle possible (model checking).

Can be used for automated planning.

Practical Importance of Temporal and Dynamic Logics:

Automatic verification in principle possible (model checking).

Can be used for automated planning.

Executable specifications can be used for programming.

Practical Importance of Temporal and Dynamic Logics:

Automatic verification in principle possible (model checking).

Can be used for automated planning.

Executable specifications can be used for programming.

Note:

When we combine time (actions) with knowledge (beliefs, desires, intentions, obligations...), we finally obtain a fairly realistic model of MAS.