

Logiques temporelles pour la vérification : expressivité, complexité, algorithmes

Soutenance de Thèse – 03 avril 2003

Nicolas MARKEY

Laboratoire d'Informatique Fondamentale d'Orléans

Formal verification?

- Why is verification **crucial**?
 - Reactive systems are **everywhere**,
 - They are ever more **complex**,
 - Numerous **bugs** have occurred (Ariane V, Therac-25, ...)

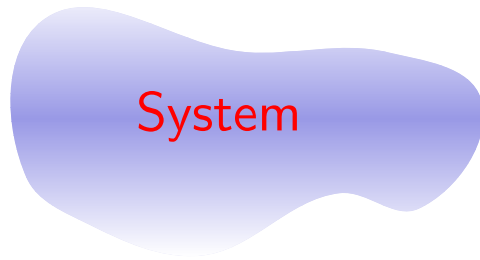
Formal verification?

- Why is verification **crucial**?
 - Reactive systems are **everywhere**,
 - They are ever more **complex**,
 - Numerous **bugs** have occurred (Ariane V, Therac-25, ...)
- Strict methods are necessary for verifying or certifying systems.

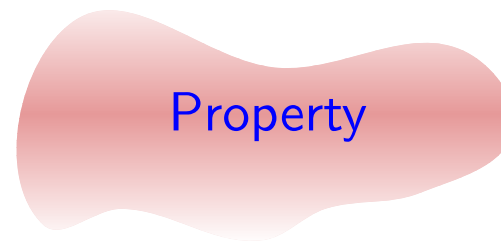
Formal verification?

- Why is verification **crucial**?
 - Reactive systems are **everywhere**,
 - They are ever more **complex**,
 - Numerous **bugs** have occurred (Ariane V, Therac-25, ...)
- Strict methods are necessary for verifying or certifying systems.
- Possible methods for **formal verification**:
 - Formal proof,
 - Testing,
 - **Model checking**...

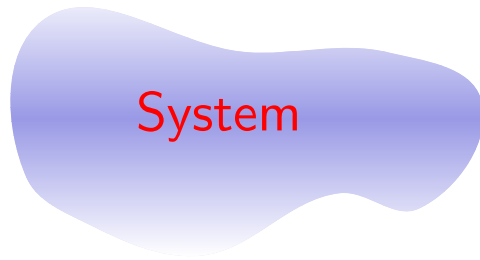
Verification by model checking



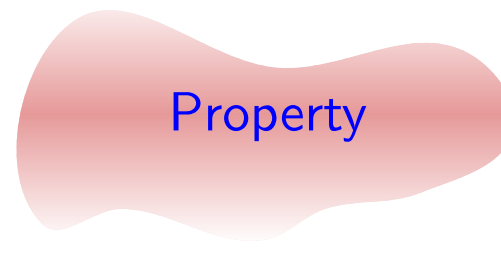
?
satisfies



Verification by model checking

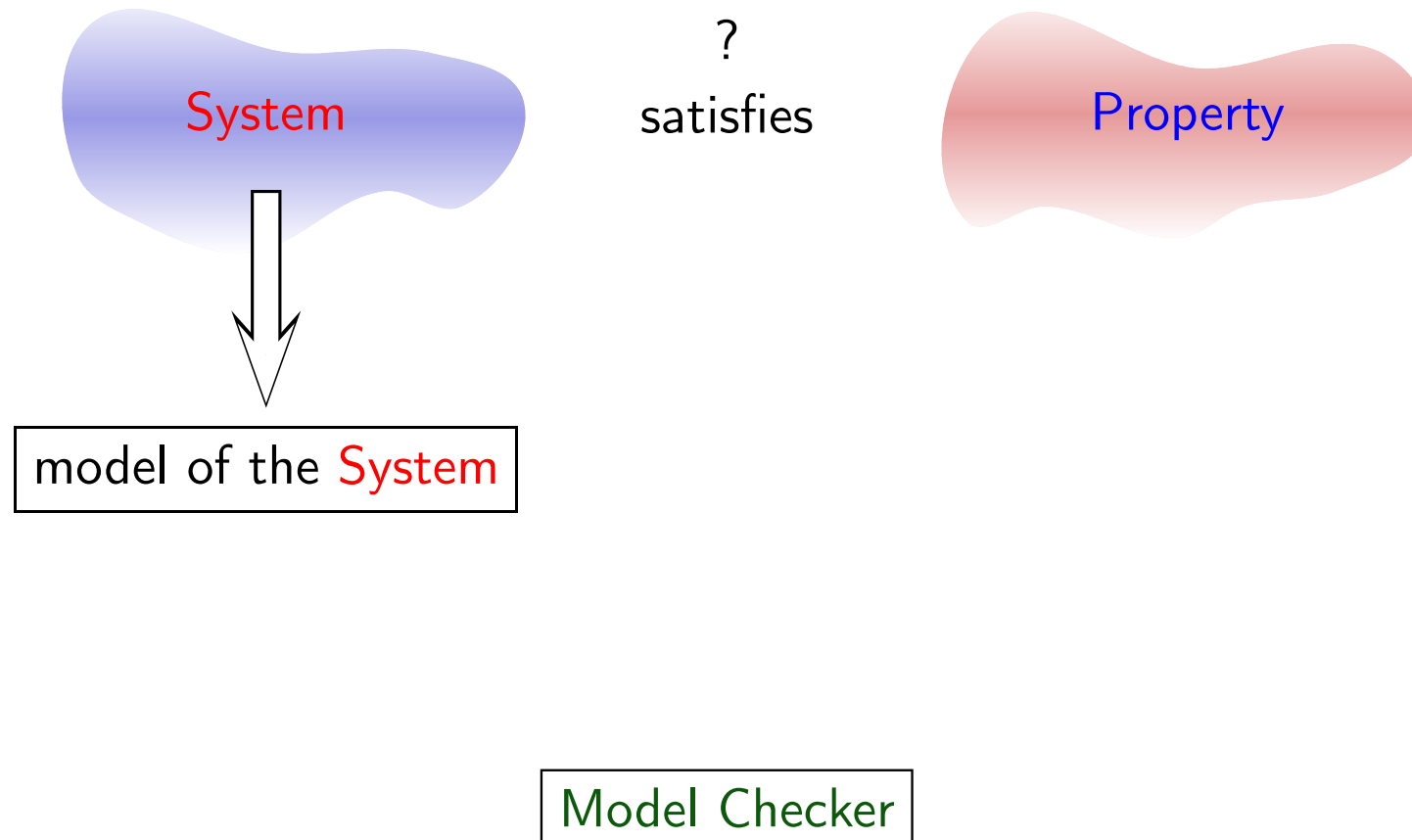


?
satisfies

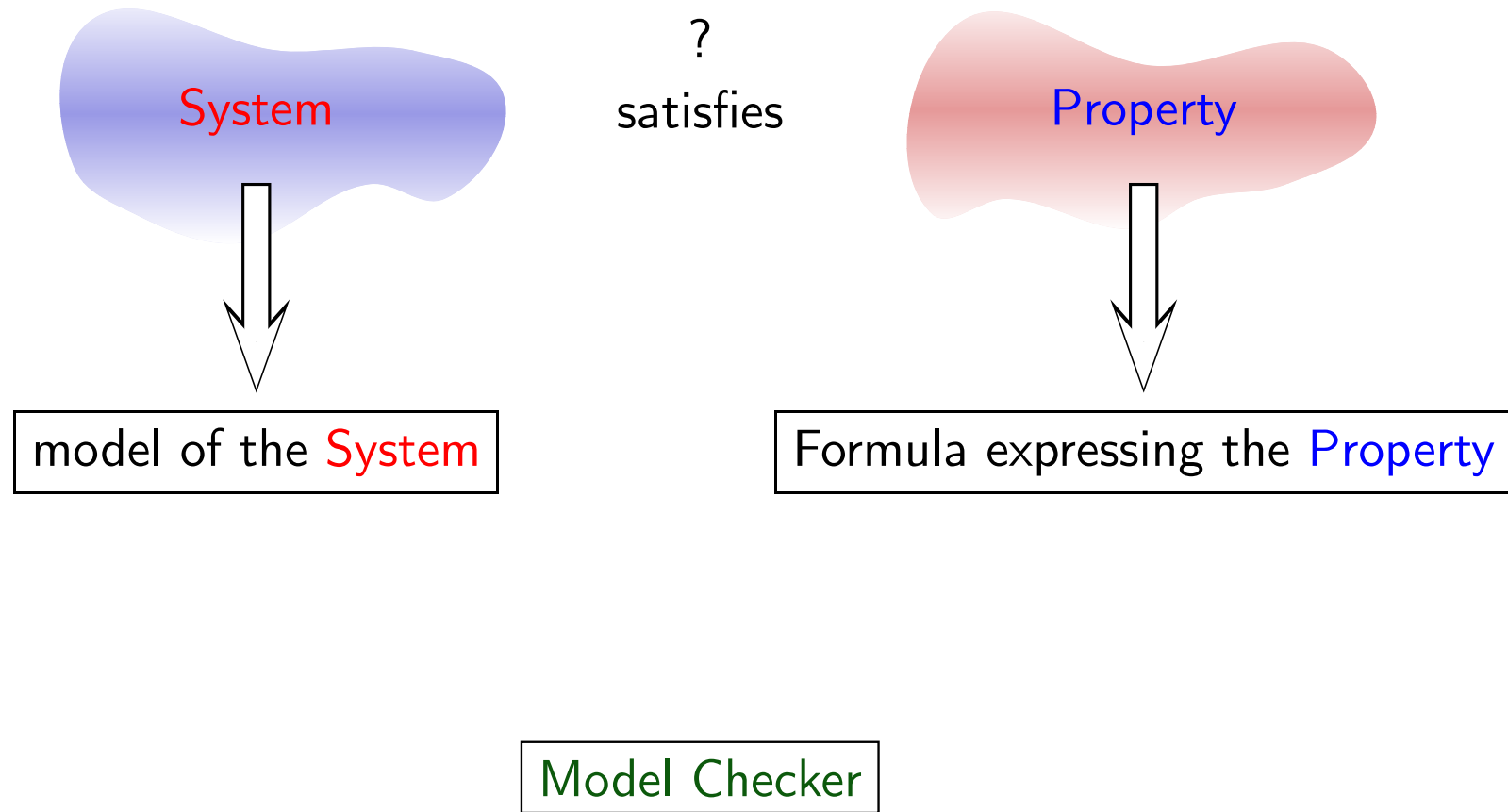


Model Checker

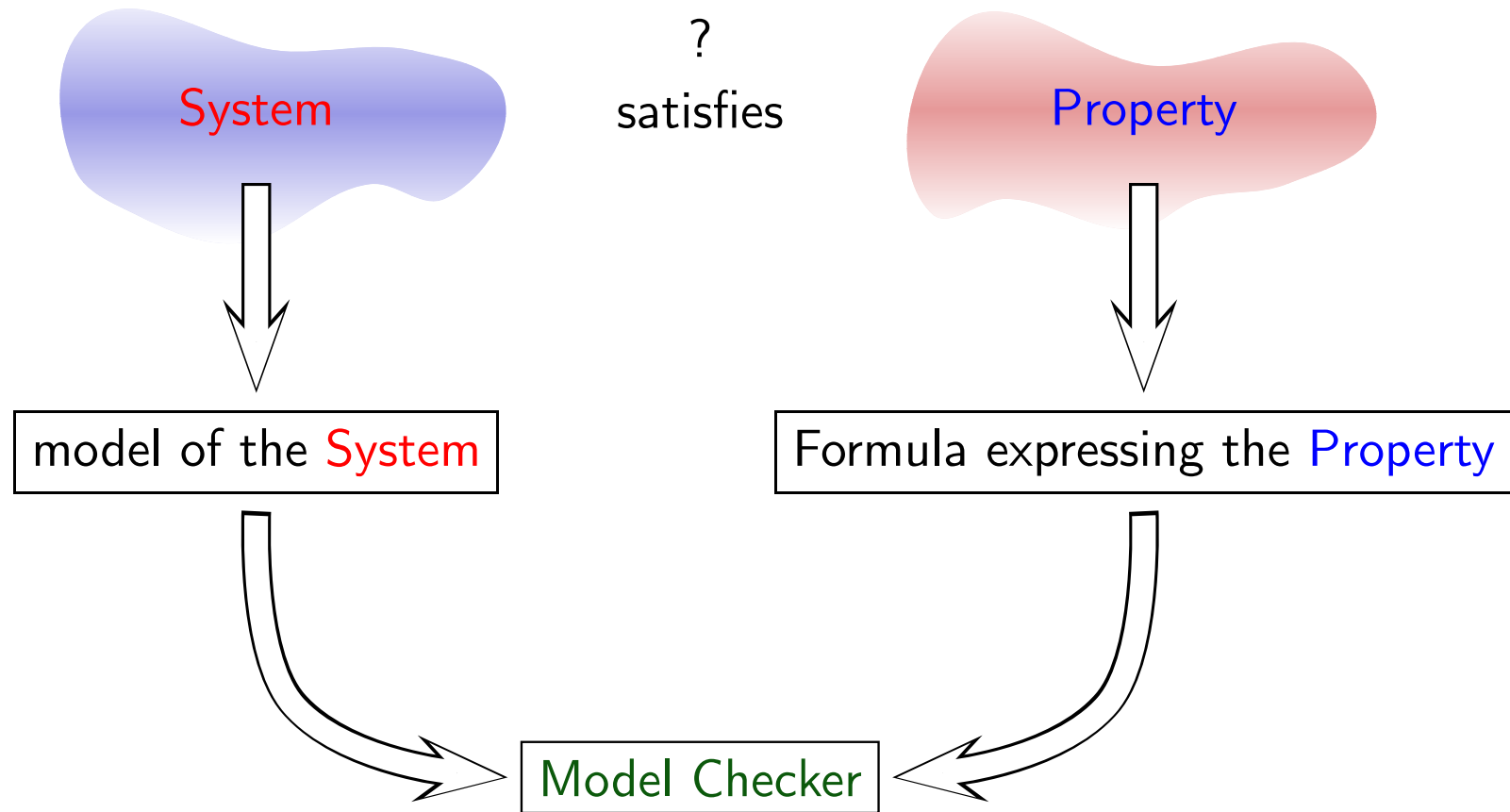
Verification by model checking



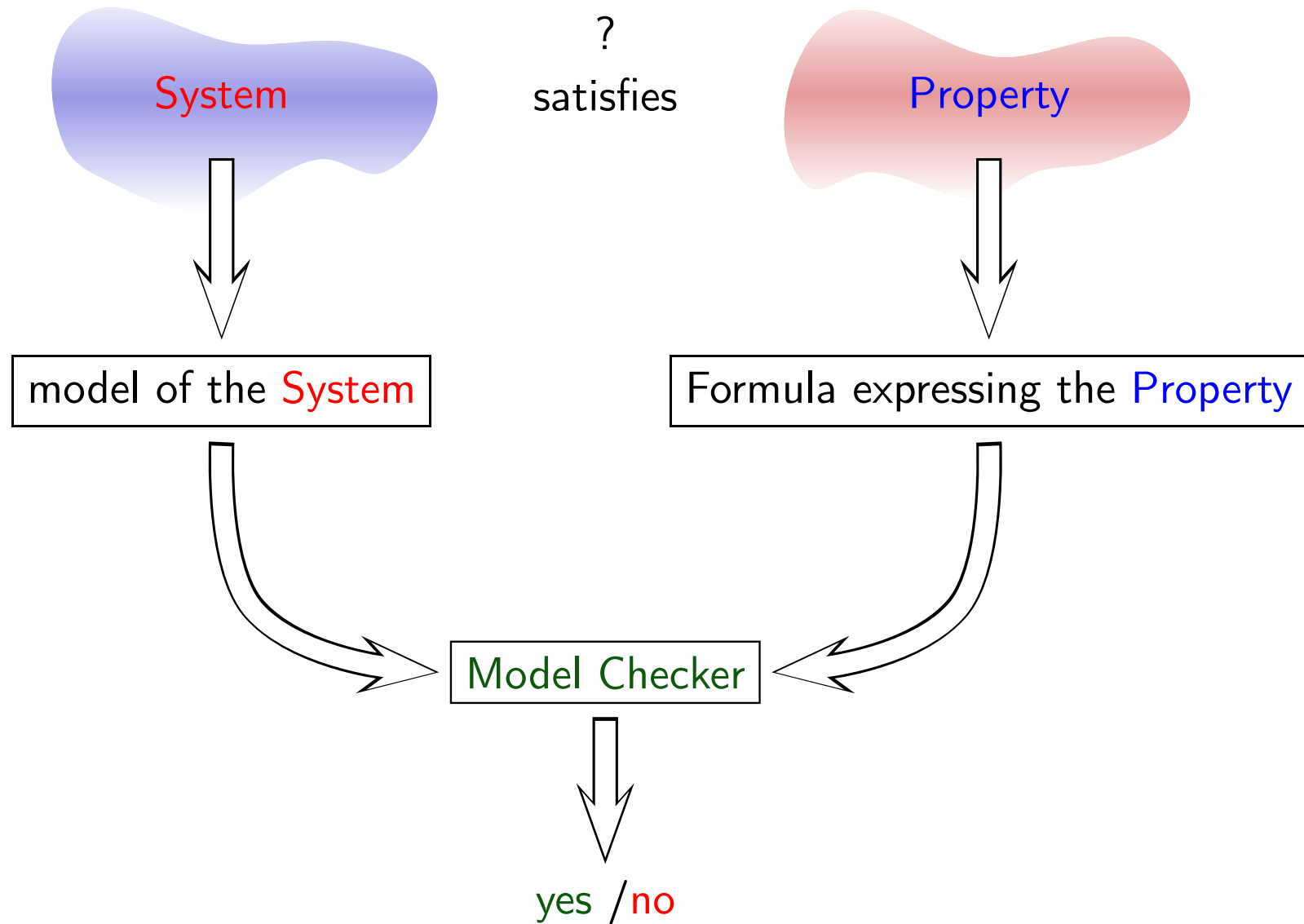
Verification by model checking



Verification by model checking



Verification by model checking



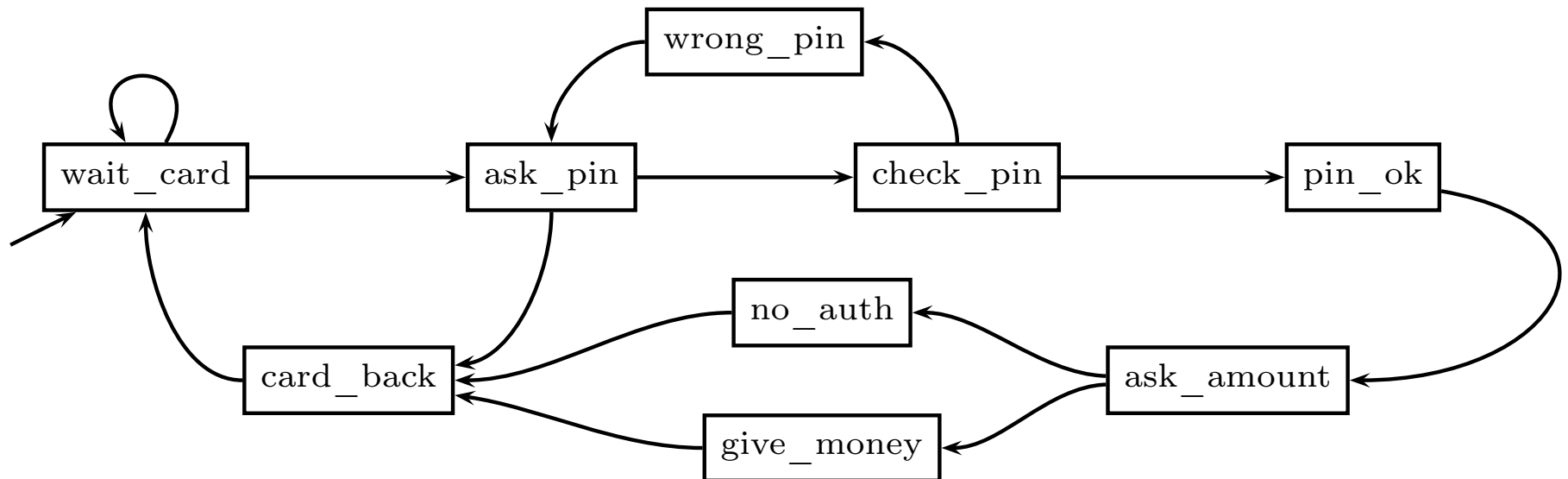
Kripke structures

We use Kripke structures for modelling the system.

Kripke structures

We use **Kripke structures** for modelling the system.

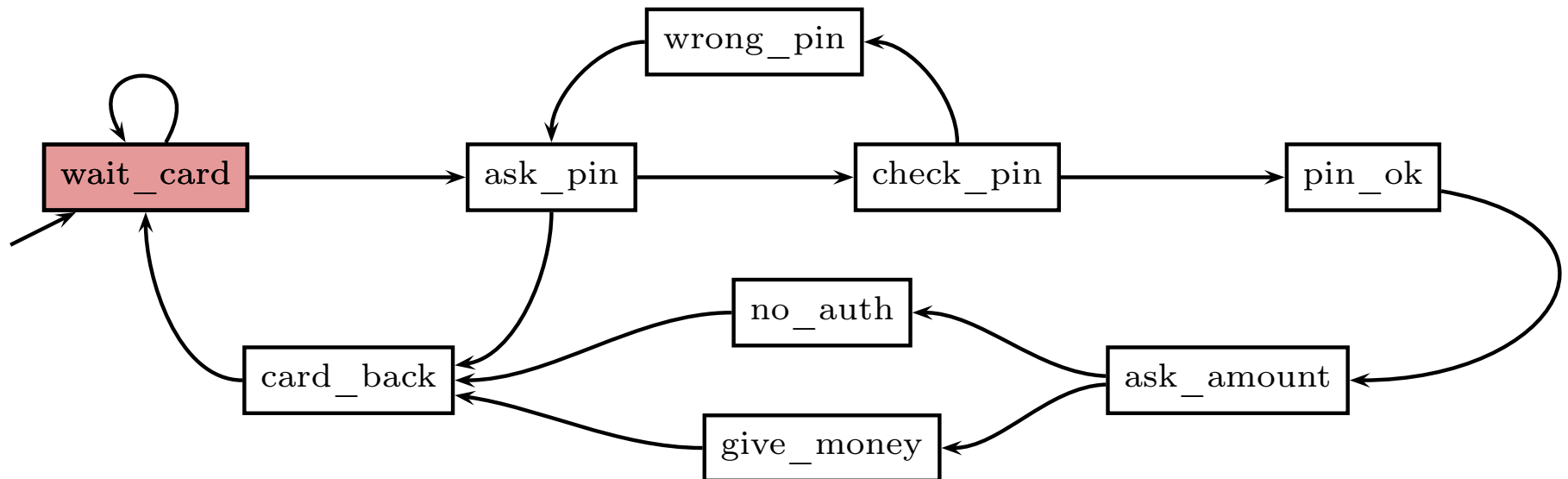
For instance, consider a (simplified) model of an A.T.M.:



Kripke structures

We use **Kripke structures** for modelling the system.

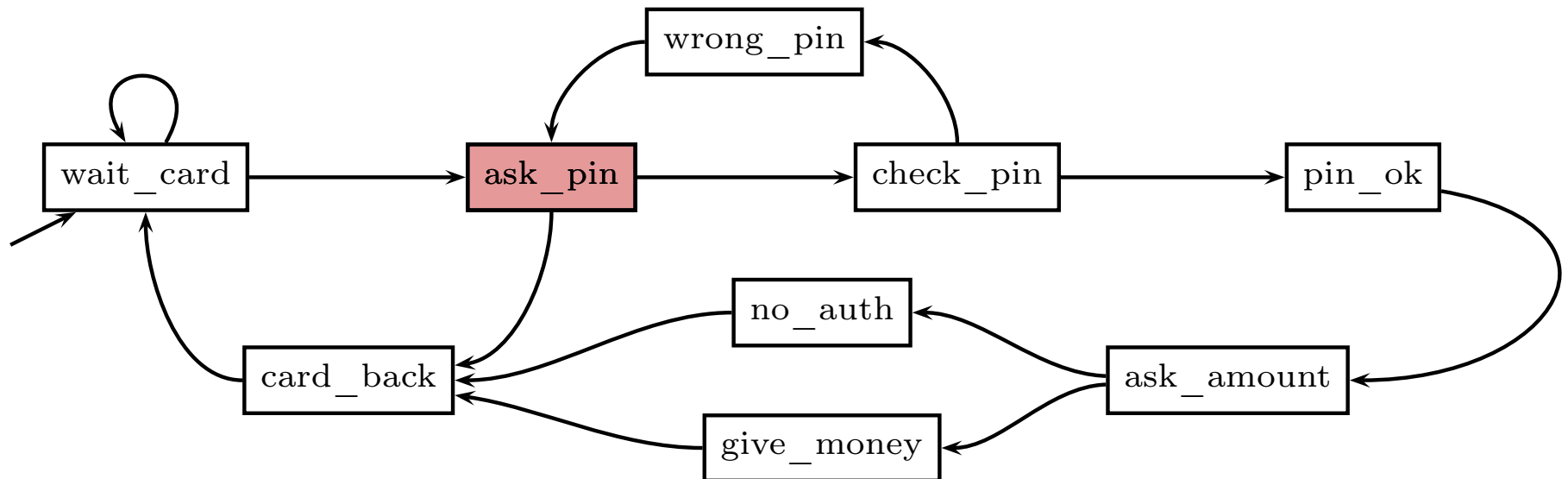
For instance, consider a (simplified) model of an A.T.M.:



Kripke structures

We use **Kripke structures** for modelling the system.

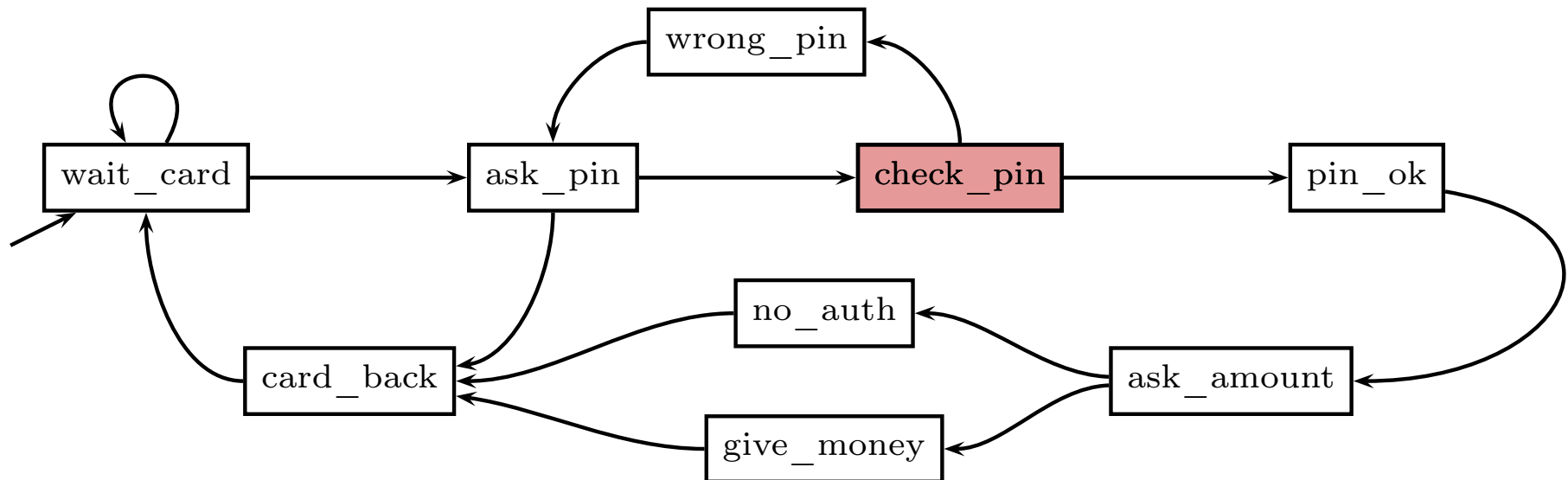
For instance, consider a (simplified) model of an A.T.M.:



Kripke structures

We use **Kripke structures** for modelling the system.

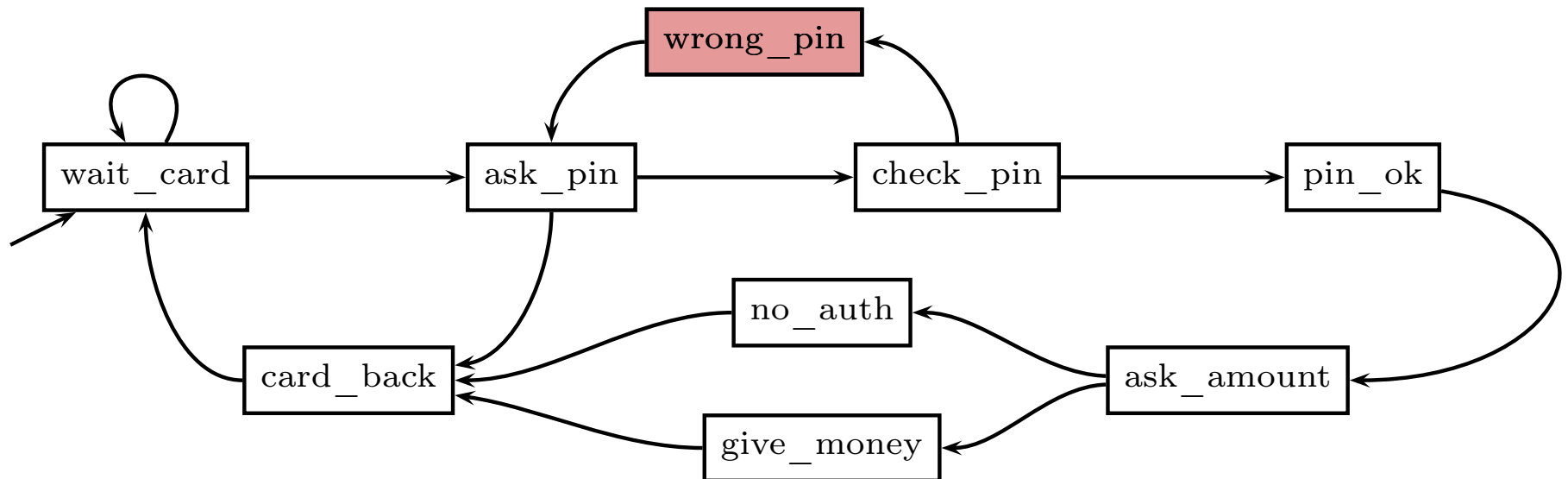
For instance, consider a (simplified) model of an A.T.M.:



Kripke structures

We use **Kripke structures** for modelling the system.

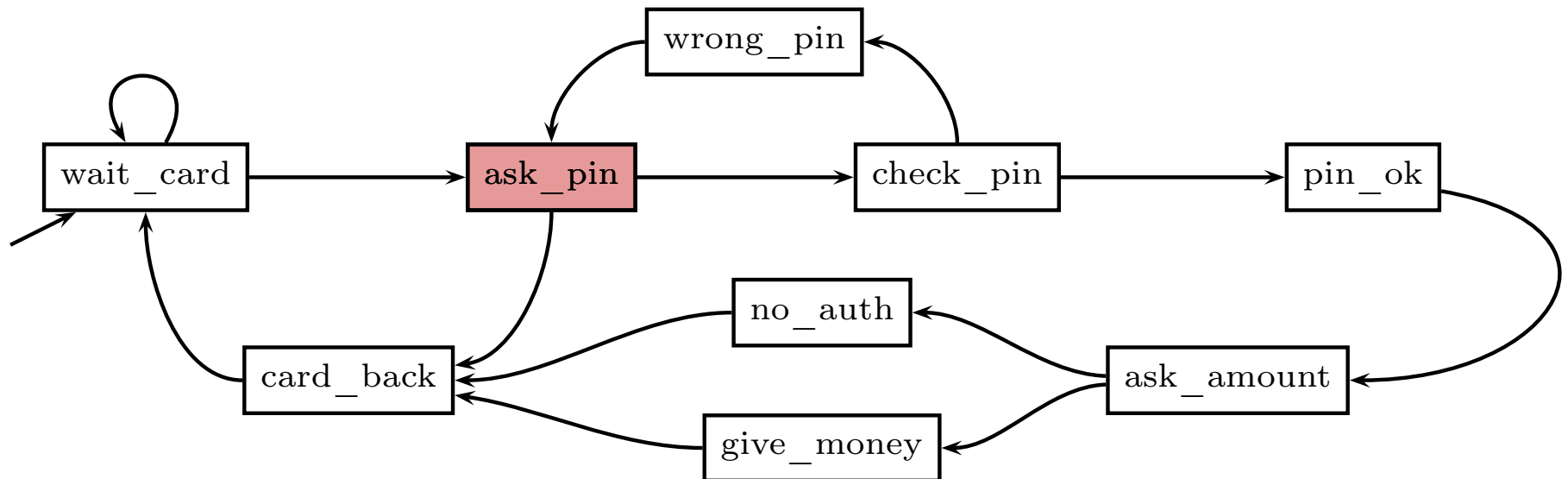
For instance, consider a (simplified) model of an A.T.M.:



Kripke structures

We use **Kripke structures** for modelling the system.

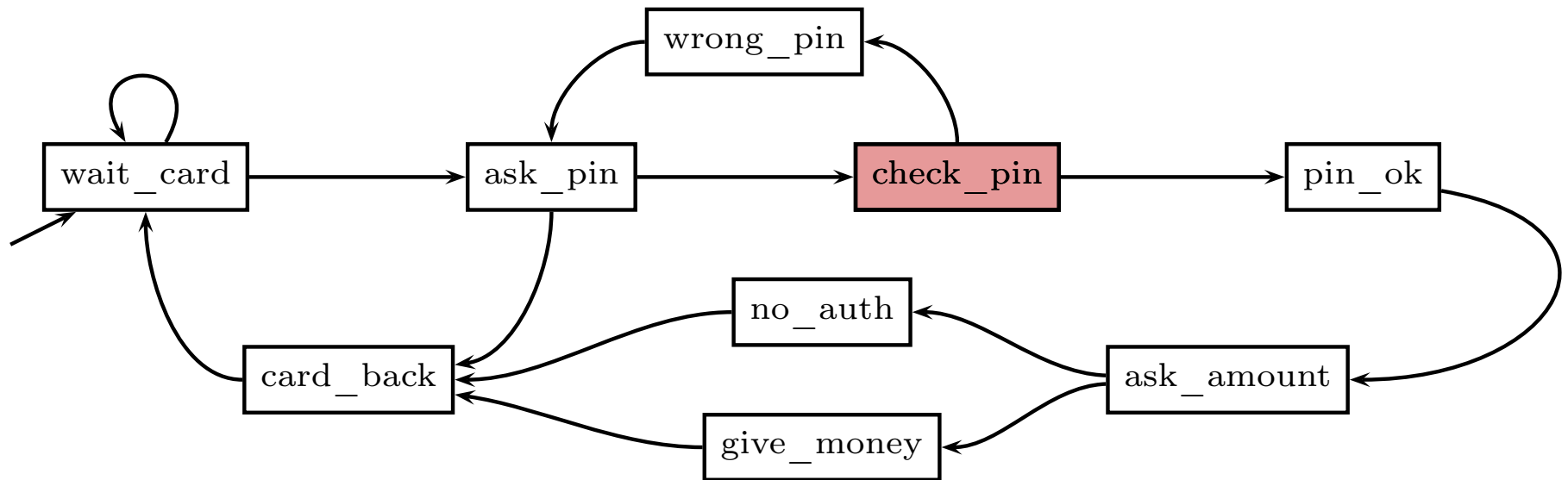
For instance, consider a (simplified) model of an A.T.M.:



Kripke structures

We use **Kripke structures** for modelling the system.

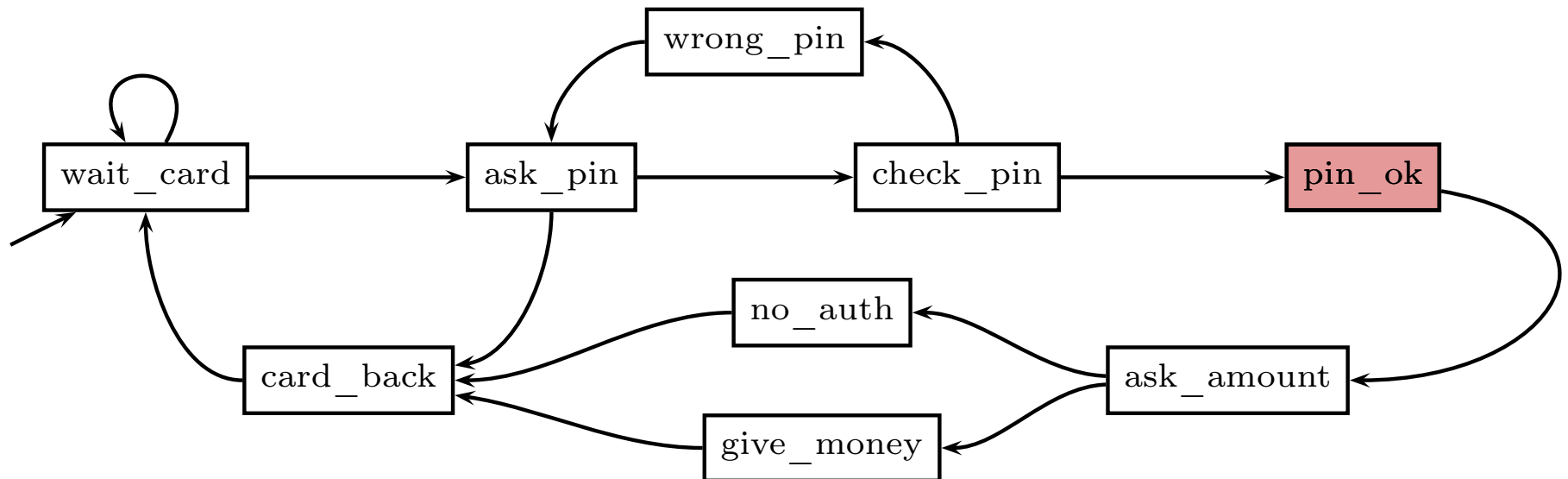
For instance, consider a (simplified) model of an A.T.M.:



Kripke structures

We use **Kripke structures** for modelling the system.

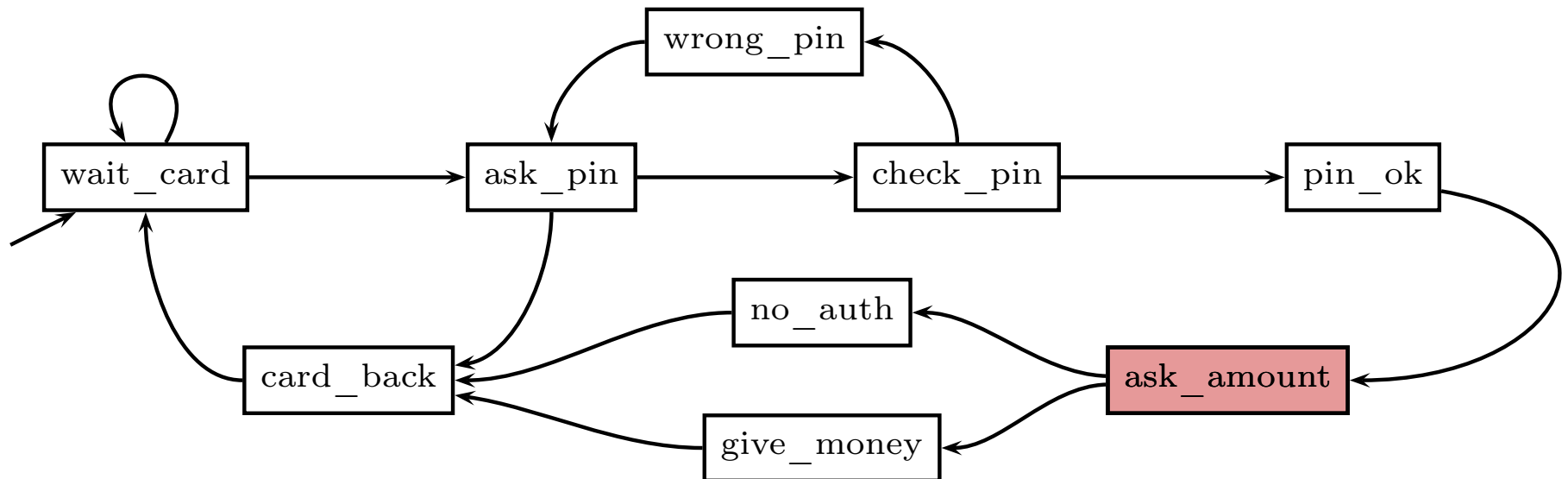
For instance, consider a (simplified) model of an A.T.M.:



Kripke structures

We use **Kripke structures** for modelling the system.

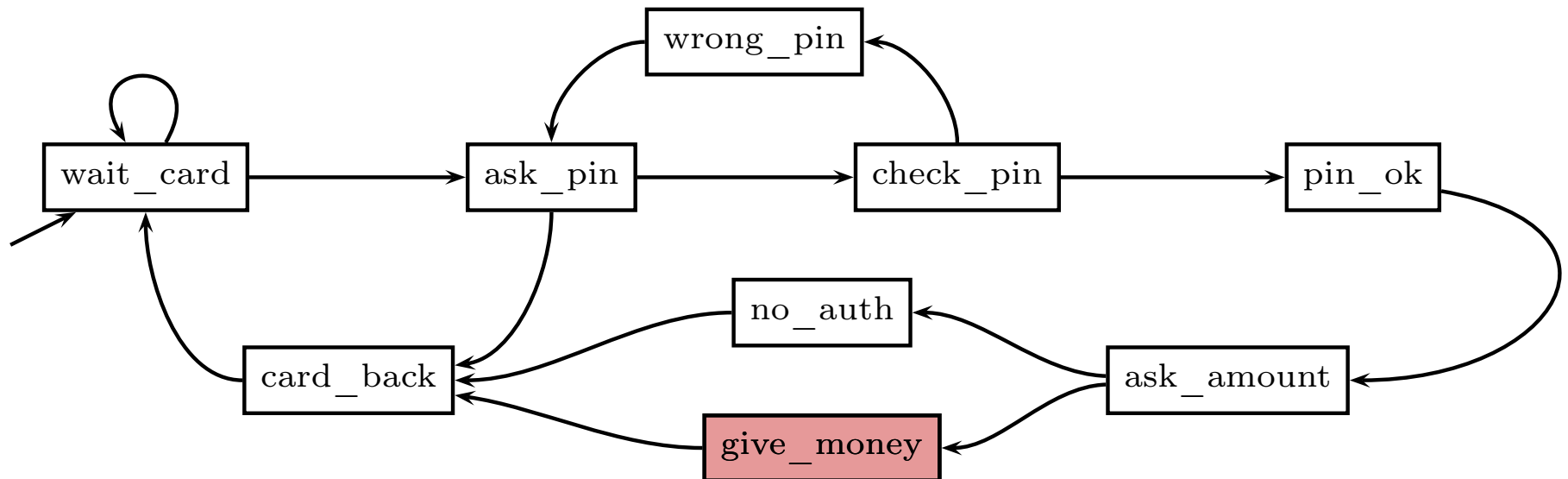
For instance, consider a (simplified) model of an A.T.M.:



Kripke structures

We use **Kripke structures** for modelling the system.

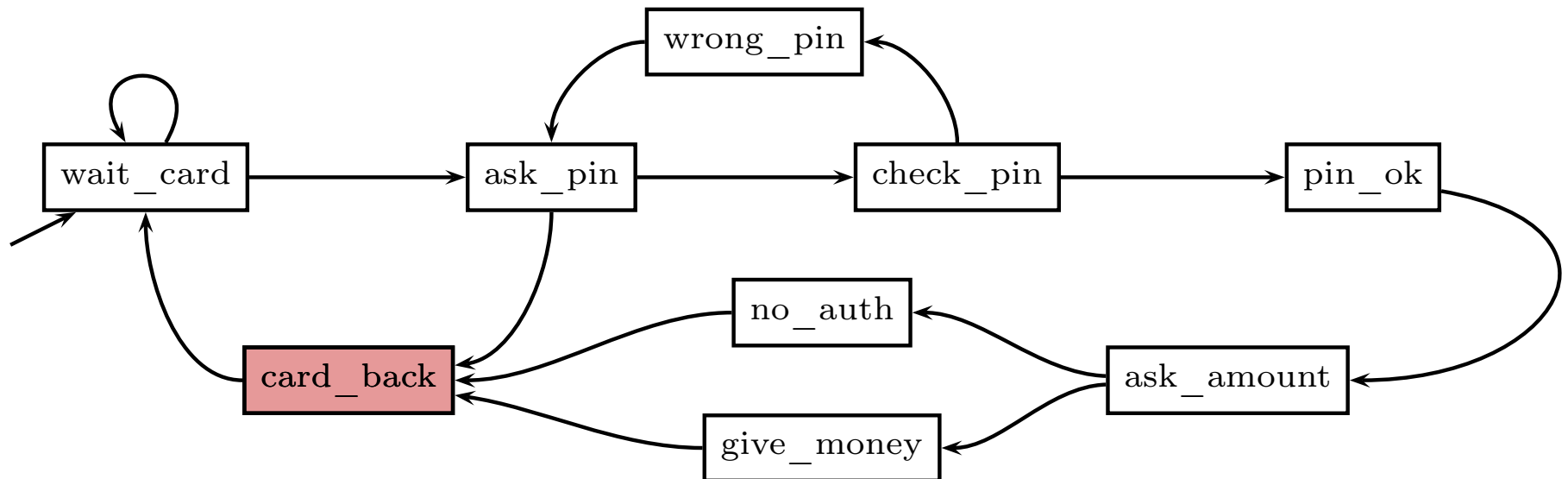
For instance, consider a (simplified) model of an A.T.M.:



Kripke structures

We use **Kripke structures** for modelling the system.

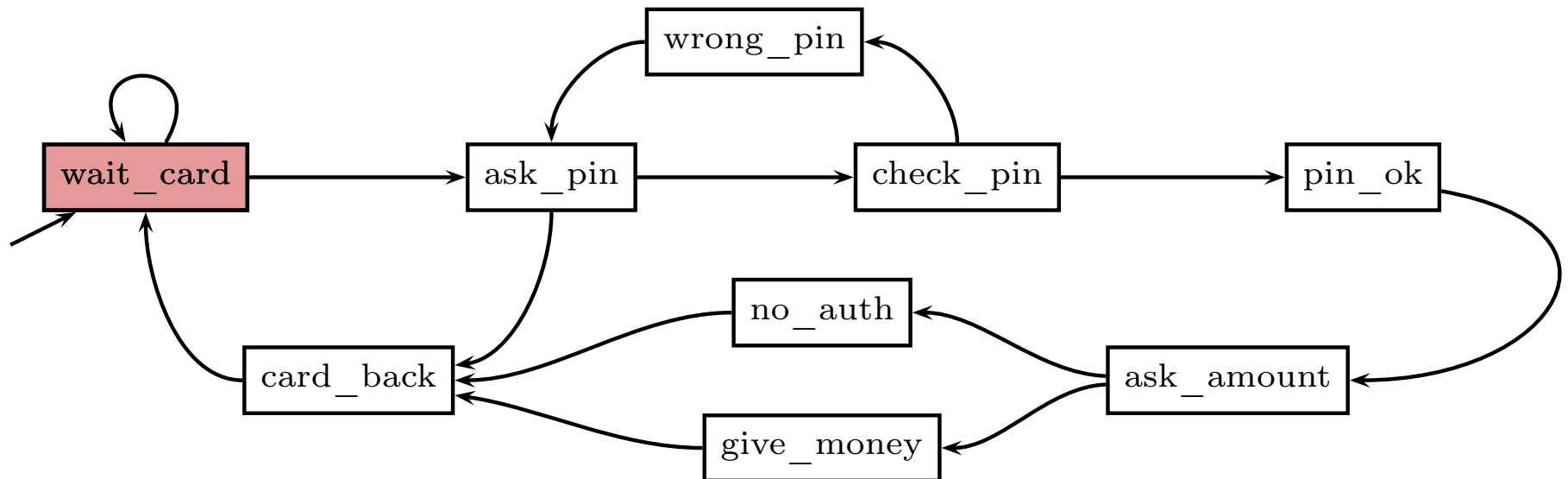
For instance, consider a (simplified) model of an A.T.M.:



Kripke structures

We use **Kripke structures** for modelling the system.

For instance, consider a (simplified) model of an A.T.M.:



Choosing a specification language

Pnueli: Using temporal logics for expressing properties.

Clarke, Sistla, Sifakis, Emerson: Model checking with temporal logics.

Lamport, Emerson: Many different temporal logics.

Choosing a specification language

Pnueli: Using temporal logics for expressing properties.

Clarke, Sistla, Sifakis, Emerson: Model checking with temporal logics.

Lamport, Emerson: Many different temporal logics.

Several criteria for comparing them:

- **Expressiveness:** Temporal logics have different expressive powers. This is an important criterion when choosing the temporal logic.

Choosing a specification language

Pnueli: Using temporal logics for expressing properties.

Clarke, Sistla, Sifakis, Emerson: Model checking with temporal logics.

Lamport, Emerson: Many different temporal logics.

Several criteria for comparing them:

- **Expressiveness:** Temporal logics have different expressive powers. This is an important criterion when choosing the temporal logic.
- **Succinctness:** Some properties can be expressed in several different temporal logics, but the formulas can be more or less long.

Choosing a specification language

Pnueli: Using temporal logics for expressing properties.

Clarke, Sistla, Sifakis, Emerson: Model checking with temporal logics.

Lamport, Emerson: Many different temporal logics.

Several criteria for comparing them:

- **Expressiveness:** Temporal logics have different expressive powers. This is an important criterion when choosing the temporal logic.
- **Succinctness:** Some properties can be expressed in several different temporal logics, but the formulas can be more or less long.
- **Complexity:** The problem of model checking a given temporal logic is more or less complex.

Outline of the talk

1. Past-time modalities in LTL

We prove that past-time modalities *do* add succinctness to **LTL**, and that they *really* don't change the complexity of model checking.

2. Extensions of CTL

We give optimal algorithms for model checking **CTL⁺**, **FCTL**, **GFCTL** and **ECTL⁺**: These problems are Δ_2^P -complete.

3. Quantitative temporal logics

We show that it is possible, in certain restricted cases, to perform *timed* model checking in **polynomial time**. We also study several other cases.

Conclusion

Outline of the talk

1. Past-time modalities in LTL

We prove that past-time modalities *do* add succinctness to **LTL**, and that they *really* don't change the complexity of model checking.

2. Extensions of CTL

We give optimal algorithms for model checking CTL^+ , FCTL, GFCTL and ECTL^+ : These problems are Δ_2^P -complete.

3. Quantitative temporal logics

We show that it is possible, in certain restricted cases, to perform *timed* model checking in polynomial time. We also study several other cases.

Conclusion

Definition of LTL + Past

LTL + Past (PLTL) is defined by the following syntax:

$$\text{PLTL} \ni \varphi, \psi ::= \neg\varphi \mid \varphi \vee \psi \mid \mathbf{X}\varphi \mid \varphi \mathbf{U}\psi \mid \mathbf{X}^{-1}\varphi \mid \varphi \mathbf{S}\psi \mid p \mid q \mid \dots$$

where p, q, \dots are atomic propositions.

Definition of LTL + Past

LTL + Past (PLTL) is defined by the following syntax:

$$\text{PLTL} \ni \varphi, \psi ::= \neg\varphi \mid \varphi \vee \psi \mid \mathbf{X}\varphi \mid \varphi \mathbf{U}\psi \mid \mathbf{X}^{-1}\varphi \mid \varphi \mathbf{S}\psi \mid p \mid q \mid \dots$$

where p, q, \dots are atomic propositions.

Some useful abbreviations:

$$\top \stackrel{\text{def}}{=} p \vee \neg p$$

$$\mathbf{F}\varphi \stackrel{\text{def}}{=} \top \mathbf{U}\varphi$$

$$\mathbf{F}^{-1}\varphi \stackrel{\text{def}}{=} \top \mathbf{S}\varphi$$

$$\mathbf{G}\varphi \stackrel{\text{def}}{=} \neg\mathbf{F}\neg\varphi$$

$$\mathbf{G}^{-1}\varphi \stackrel{\text{def}}{=} \neg\mathbf{F}^{-1}\neg\varphi$$

$$\mathbf{F}^{\infty}\varphi \stackrel{\text{def}}{=} \mathbf{G}\mathbf{F}\varphi$$

$$\mathbf{I}\varphi \stackrel{\text{def}}{=} \mathbf{G}^{-1}\mathbf{F}^{-1}\varphi$$

$$\mathbf{G}^{\infty}\varphi \stackrel{\text{def}}{=} \mathbf{F}\mathbf{G}\varphi$$

Definition of LTL + Past

LTL + Past (PLTL) is defined by the following syntax:

$$\text{PLTL} \ni \varphi, \psi ::= \neg\varphi \mid \varphi \vee \psi \mid \mathbf{X}\varphi \mid \varphi \mathbf{U}\psi \mid \mathbf{X}^{-1}\varphi \mid \varphi \mathbf{S}\psi \mid p \mid q \mid \dots$$

where p, q, \dots are atomic propositions.

Some useful abbreviations:

$$\top \stackrel{\text{def}}{=} p \vee \neg p$$

$$\mathbf{F}\varphi \stackrel{\text{def}}{=} \top \mathbf{U}\varphi$$

$$\mathbf{F}^{-1}\varphi \stackrel{\text{def}}{=} \top \mathbf{S}\varphi$$

$$\mathbf{G}\varphi \stackrel{\text{def}}{=} \neg\mathbf{F}\neg\varphi$$

$$\mathbf{G}^{-1}\varphi \stackrel{\text{def}}{=} \neg\mathbf{F}^{-1}\neg\varphi$$

$$\mathbf{F}^{\infty}\varphi \stackrel{\text{def}}{=} \mathbf{G}\mathbf{F}\varphi$$

$$\mathbf{I}\varphi \stackrel{\text{def}}{=} \mathbf{G}^{-1}\mathbf{F}^{-1}\varphi$$

$$\mathbf{G}^{\infty}\varphi \stackrel{\text{def}}{=} \mathbf{F}\mathbf{G}\varphi$$

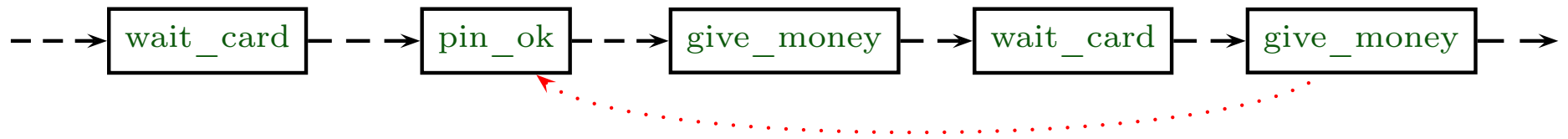
Example:

$$\mathbf{G}(\text{give_money} \Rightarrow \mathbf{F}^{-1}\text{pin_ok})$$

Forgettable past [LS95]

$$G(\text{give_money} \Rightarrow \mathbf{F}^{-1} \text{pin_ok})$$

This is not what we want to express: The following path satisfies the formula:

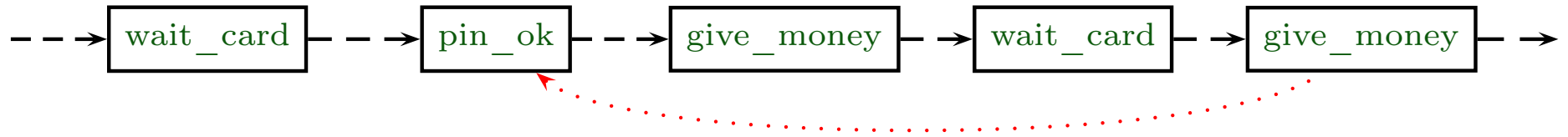


“ $\mathbf{F}^{-1} \text{pin_ok}$ ” should only refer to what happened since the latest `wait_card`.

Forgettable past [LS95]

$$\mathbf{G} (\text{give_money} \Rightarrow \mathbf{F}^{-1} \text{pin_ok})$$

This is not what we want to express: The following path satisfies the formula:



“ $\mathbf{F}^{-1} \text{pin_ok}$ ” should only refer to what happened since the latest `wait_card`.

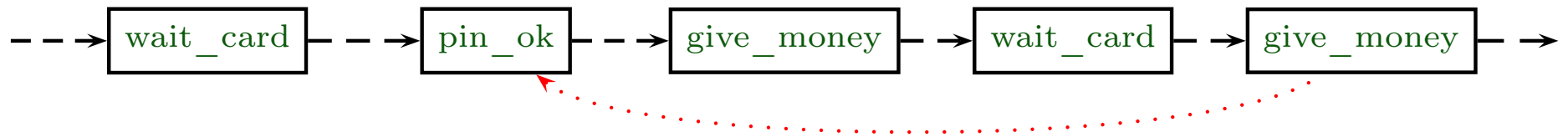
Operator for “forgetting the past”: \mathbf{N} (from now on):

$$\pi, i \models \mathbf{N} \varphi$$

Forgettable past [LS95]

$$\mathbf{G} (\text{give_money} \Rightarrow \mathbf{F}^{-1} \text{pin_ok})$$

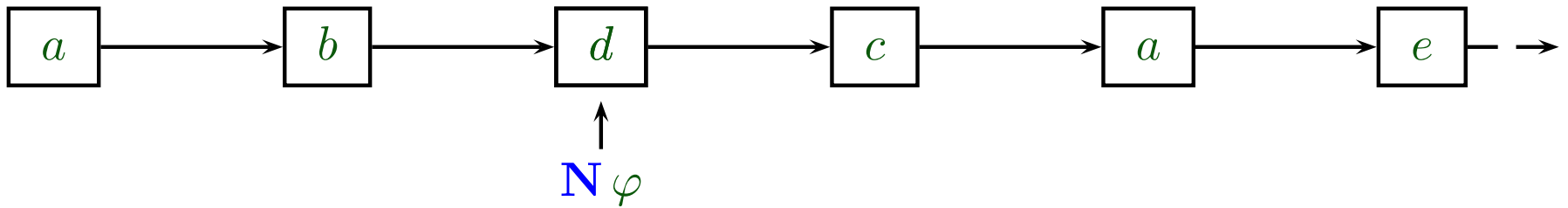
This is not what we want to express: The following path satisfies the formula:



“ $\mathbf{F}^{-1} \text{pin_ok}$ ” should only refer to what happened since the latest `wait_card`.

Operator for “forgetting the past”: \mathbf{N} (from now on):

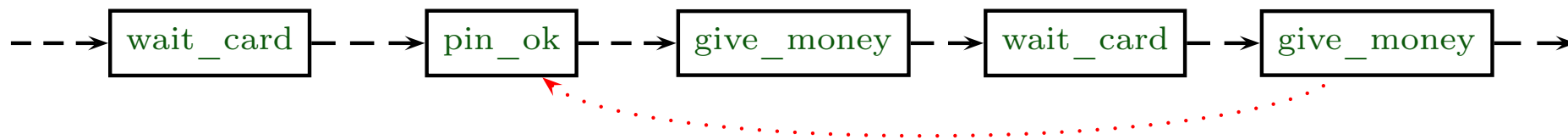
$$\pi, i \models \mathbf{N} \varphi$$



Forgettable past [LS95]

$$\mathbf{G} (\text{give_money} \Rightarrow \mathbf{F}^{-1} \text{pin_ok})$$

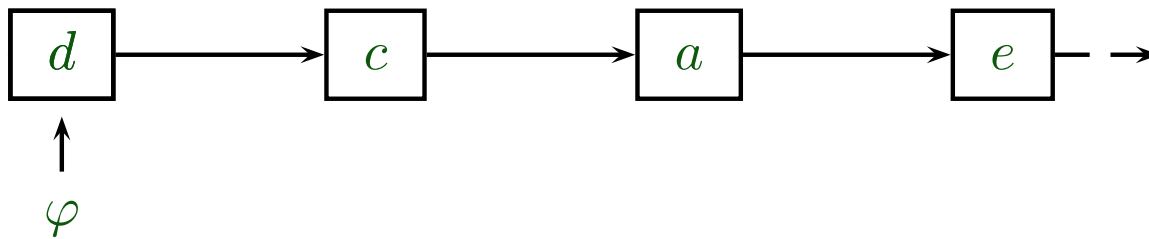
This is not what we want to express: The following path satisfies the formula:



“ $\mathbf{F}^{-1} \text{pin_ok}$ ” should only refer to what happened since the latest `wait_card`.

Operator for “forgetting the past”: \mathbf{N} (from now on):

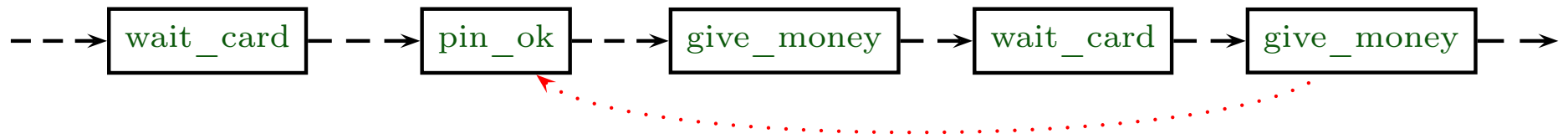
$$\pi, i \models \mathbf{N} \varphi \quad \Leftrightarrow \quad \pi^{\geq i}, 0 \models \varphi$$



Forgettable past [LS95]

$$\mathbf{G} (\text{give_money} \Rightarrow \mathbf{F}^{-1} \text{pin_ok})$$

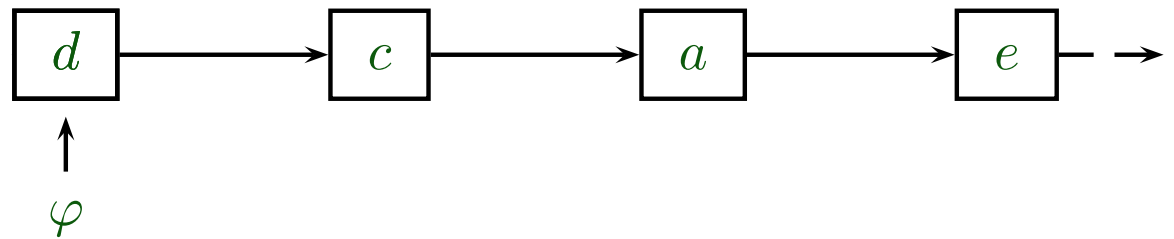
This is not what we want to express: The following path satisfies the formula:



“ $\mathbf{F}^{-1} \text{pin_ok}$ ” should only refer to what happened since the latest `wait_card`.

Operator for “forgetting the past”: \mathbf{N} (from now on):

$$\pi, i \models \mathbf{N} \varphi \quad \Leftrightarrow \quad \pi^{\geq i}, 0 \models \varphi$$



Example:

$$\mathbf{G} (\text{wait_card} \Rightarrow \mathbf{N} \mathbf{G} (\text{give_money} \Rightarrow \mathbf{F}^{-1} \text{pin_ok}))$$

Expressive power

$$\mathbf{G} (a \Rightarrow \mathbf{F}^{-1} b) \equiv_i \neg((\neg b) \mathbf{U} (a \wedge \neg b))$$

Expressive power

$$\mathbf{G} (a \Rightarrow \mathbf{F}^{-1} b) \equiv_i \neg((\neg b) \mathbf{U} (a \wedge \neg b))$$

Theorem [Kam68,GPSS80]: PLTL and LTL have the same expressive power.

Corollary [LMS02]: NLTL and LTL have the same expressive power.

Expressive power

$$\mathbf{G} (a \Rightarrow \mathbf{F}^{-1} b) \equiv_i \neg((\neg b) \mathbf{U} (a \wedge \neg b))$$

Theorem [Kam68,GPSS80]: PLTL and LTL have the same expressive power.

Corollary [LMS02]: NLTL and LTL have the same expressive power.

[Gab87] provides an effective algorithm for translating a PLTL formula into an (initially) equivalent LTL formula.

Expressive power

$$\mathbf{G} (a \Rightarrow \mathbf{F}^{-1} b) \equiv_i \neg((\neg b) \mathbf{U} (a \wedge \neg b))$$

Theorem [Kam68,GPSS80]: PLTL and LTL have the same expressive power.

Corollary [LMS02]: NLTL and LTL have the same expressive power.

[Gab87] provides an effective algorithm for translating a PLTL formula into an (initially) equivalent LTL formula.

The LTL equivalent formula is much less intuitive. Moreover, the best known translation involves a **triple-exponential** increase in the size of the formula.

Expressive power

$$\mathbf{G} (a \Rightarrow \mathbf{F}^{-1} b) \equiv_i \neg((\neg b) \mathbf{U} (a \wedge \neg b))$$

Theorem [Kam68,GPSS80]: PLTL and LTL have the same expressive power.

Corollary [LMS02]: NLTL and LTL have the same expressive power.

[Gab87] provides an effective algorithm for translating a PLTL formula into an (initially) equivalent LTL formula.

The LTL equivalent formula is much less intuitive. Moreover, the best known translation involves a **triple-exponential** increase in the size of the formula.

Can we avoid this explosion?

Succinctness of PLTL

Theorem [LMS02]: PLTL can be exponentially more succinct than LTL.

Succinctness of PLTL

Theorem [LMS02]: PLTL can be exponentially more succinct than LTL.

Proof: Let $\{p_0, p_1, \dots, p_n\}$ be a set of atomic propositions.

Succinctness of PLTL

Theorem [LMS02]: PLTL can be exponentially more succinct than LTL.

Proof: Let $\{p_0, p_1, \dots, p_n\}$ be a set of atomic propositions.

The PLTL formula

$$\Phi \stackrel{\text{def}}{=} \mathbf{G} \left[\left(\bigwedge_{i=1}^n (p_i \Leftrightarrow \mathbf{I} p_i) \right) \Rightarrow (p_0 \Leftrightarrow \mathbf{I} p_0) \right]$$

states that “any future state that agrees with the initial state on p_1, \dots, p_n also agrees on p_0 ”.

Succinctness of PLTL

Theorem [LMS02]: PLTL can be exponentially more succinct than LTL.

Proof: Let $\{p_0, p_1, \dots, p_n\}$ be a set of atomic propositions.

The PLTL formula

$$\Phi \stackrel{\text{def}}{=} \mathbf{G} \left[\left(\bigwedge_{i=1}^n (p_i \Leftrightarrow \mathbf{I} p_i) \right) \Rightarrow (p_0 \Leftrightarrow \mathbf{I} p_0) \right]$$

states that “any future state that agrees with the initial state on p_1, \dots, p_n also agrees on p_0 ”.

Let Ψ be an LTL formula initially equivalent to Φ .

Therefore $\mathbf{G} \Psi$ expresses the following property:

“any two future states that agree on p_1, \dots, p_n also agree on p_0 ”

Succinctness of PLTL

Theorem [LMS02]: PLTL can be exponentially more succinct than LTL.

Proof: Let $\{p_0, p_1, \dots, p_n\}$ be a set of atomic propositions.

The PLTL formula

$$\Phi \stackrel{\text{def}}{=} \mathbf{G} \left[\left(\bigwedge_{i=1}^n (p_i \Leftrightarrow \mathbf{I} p_i) \right) \Rightarrow (p_0 \Leftrightarrow \mathbf{I} p_0) \right]$$

states that “any future state that agrees with the initial state on p_1, \dots, p_n also agrees on p_0 ”.

Let Ψ be an LTL formula initially equivalent to Φ .

Therefore $\mathbf{G} \Psi$ expresses the following property:

“any two future states that agree on p_1, \dots, p_n also agree on p_0 ”

Any Büchi automaton recognizing that property has at least 2^{2^n} states. [EVW97]

The size of any LTL (or even PLTL) formula expressing that property is in $\Omega(2^n)$.

Succinctness of NLTL

Theorem [LMS02]: NLTL can be exponentially more succinct than PLTL.

Succinctness of NLTL

Theorem [LMS02]: NLTL can be exponentially more succinct than PLTL.

Proof:

We still write

$$\Phi \stackrel{\text{def}}{=} \mathbf{G} \left[\left(\bigwedge_{i=1}^n (p_i \Leftrightarrow \mathbf{I} p_i) \right) \Rightarrow (p_0 \Leftrightarrow \mathbf{I} p_0) \right]$$

The NLTL formula $\mathbf{G} \mathbf{N} \Phi$ clearly states that “any two future states that agree on p_1, \dots, p_n also agree on p_0 ”.

The size of any equivalent PLTL formula is in $\Omega(2^n)$.

Model checking fragments of NLTL

Model checking:

Given φ and a Kripke structure K , do we have, for any run π of K : $\pi, 0 \models \varphi$?

Model checking fragments of NLTL

Model checking:

Given φ and a Kripke structure K , do we have, for any run π of K : $\pi, 0 \models \varphi$?

[SC85] proves that model checking is PSPACE-complete for LTL and PLTL.

Is past *always* for free?

Model checking fragments of NLTL

Model checking:

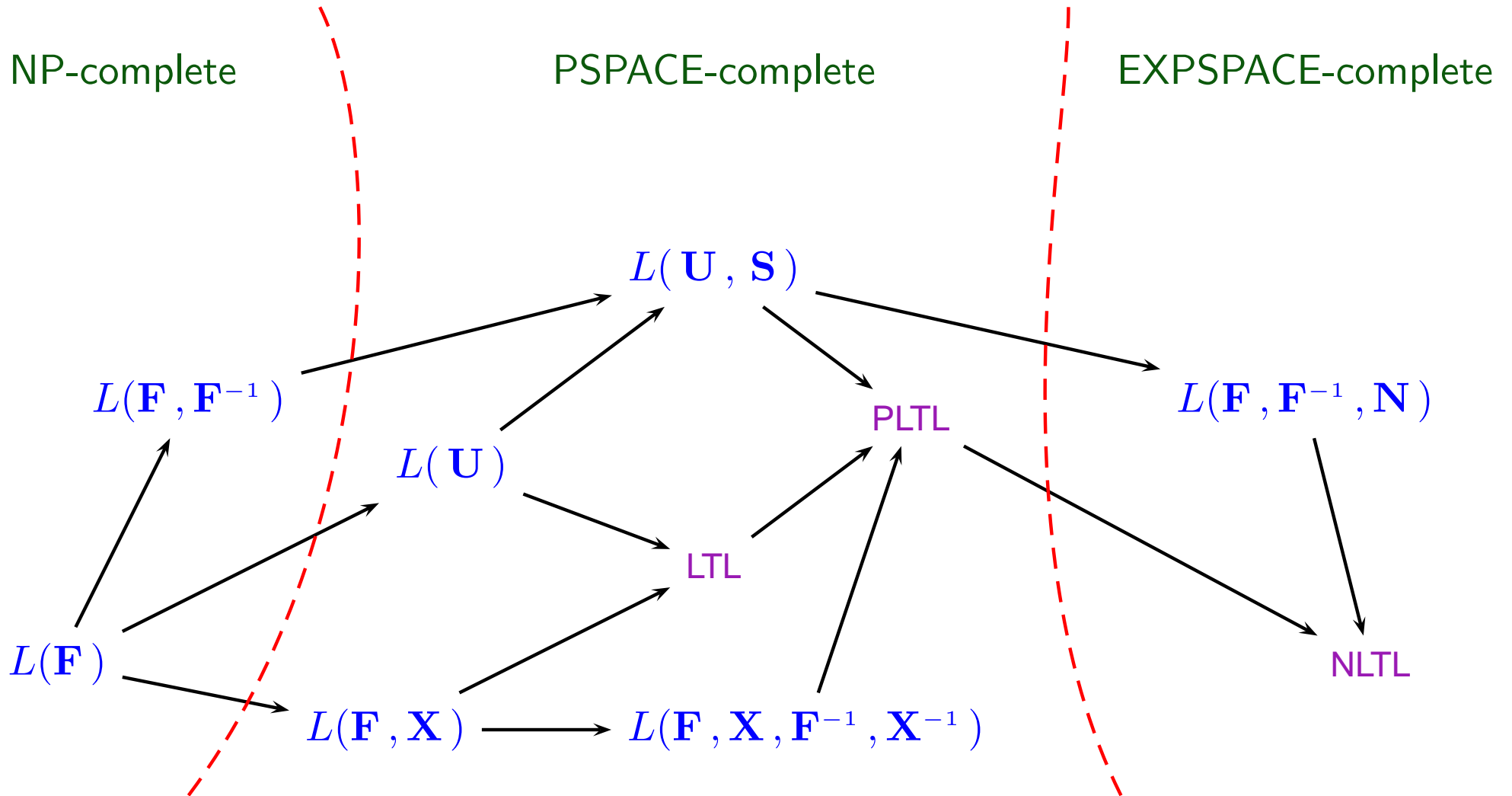
Given φ and a Kripke structure K , do we have, for any run π of K : $\pi, 0 \models \varphi$?

[SC85] proves that model checking is PSPACE-complete for LTL and PLTL.

Is past *always* for free?

\rightsquigarrow complexity of fragments of NLTL.

Complexity of fragments of NLTL



Outline of the talk

1. Past-time modalities in LTL

We prove that past-time modalities *do* add succinctness to LTL, and that they *really* don't change the complexity of model checking.

2. Extensions of CTL

We give optimal algorithms for model checking CTL^+ , FCTL , GFCTL and ECTL^+ : These problems are Δ_2^P -complete.

3. Quantitative temporal logics

We show that it is possible, in certain restricted cases, to perform *timed* model checking in polynomial time. We also study several other cases.

Conclusion

Branching-time temporal logics

CTL

CTL [CE81, QS82]: Path quantification for all temporal modalities

Model checking is P-complete

Example: **AG** (**EF** card_back)

Branching-time temporal logics

CTL \longrightarrow CTL*

CTL* [EH86]: Path quantification independent of temporal modalities
Model checking is PSPACE-complete [CES86]

Example: $\mathbf{AF}(\text{ask_pin} \wedge \mathbf{X} \text{check_pin})$

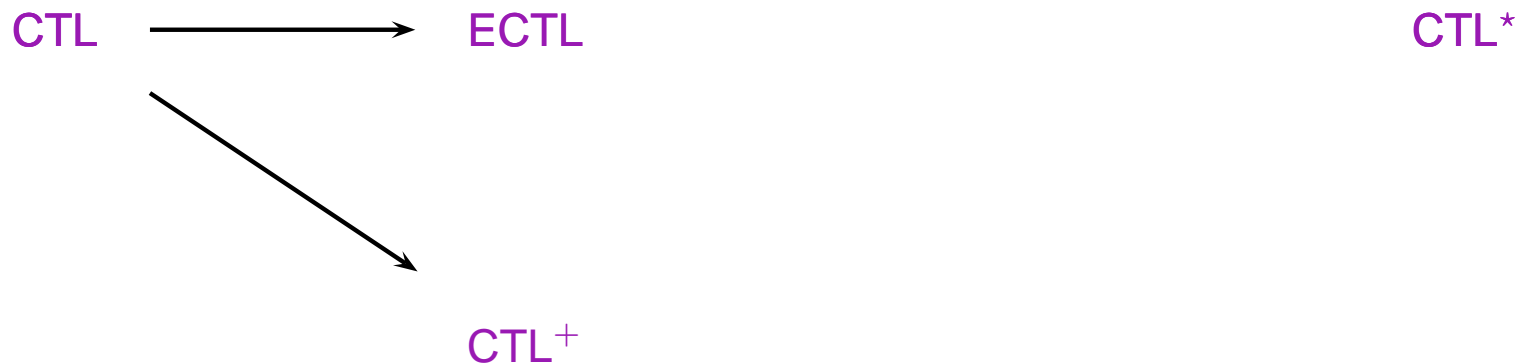
Branching-time temporal logics

CTL \longrightarrow ECTL

CTL*

ECTL [EH86]: Allows $\mathbf{E \bar{F}^\infty}$ and $\mathbf{A \bar{F}^\infty}$
Strictly more expressive than CTL
Model checking is P-complete
Example: $\mathbf{EF^\infty}$ (give_money)

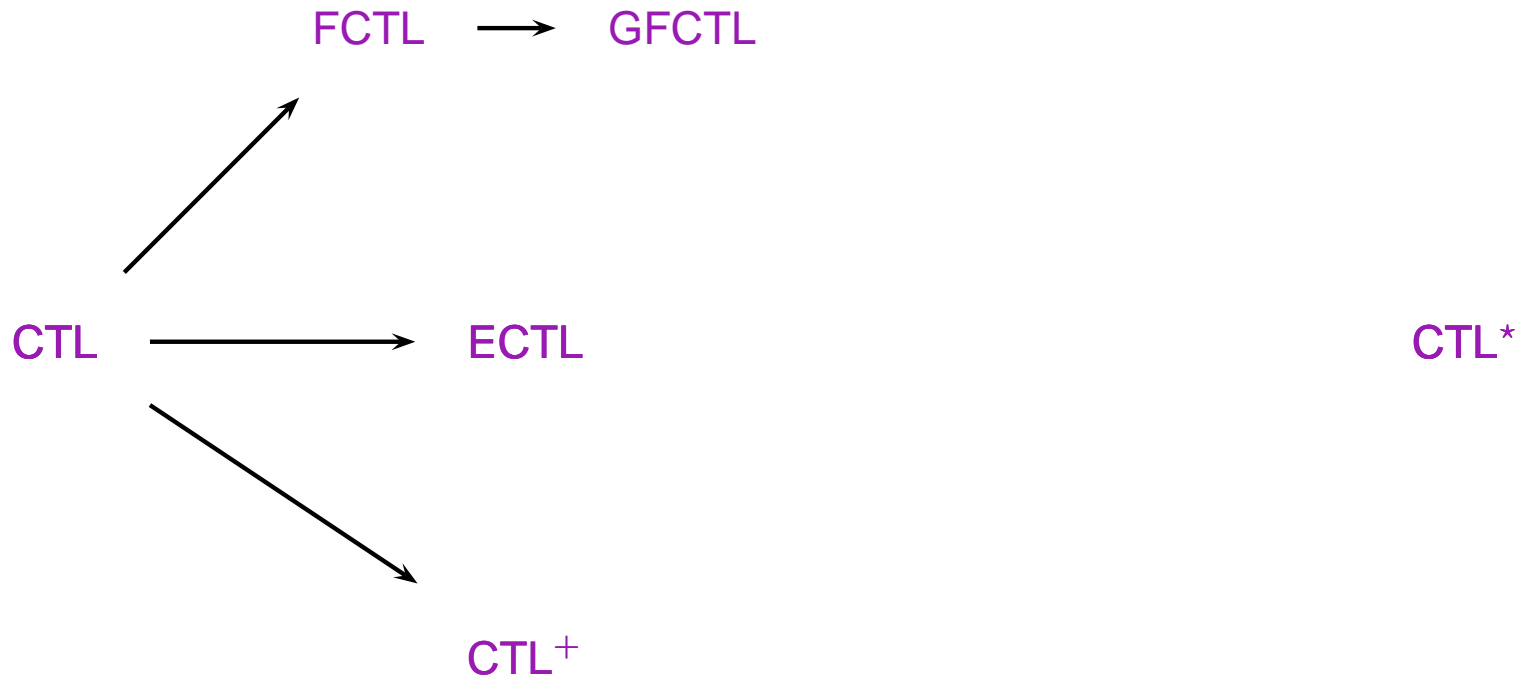
Branching-time temporal logics



CTL⁺ [EH85]: Boolean combinations in the scope of path quantifiers
Not more expressive than **CTL**, but exponentially more succinct [Wil99,AI01]
Model checking is **NP-hard** [CES86]

Example: $\mathbf{E}(\mathbf{G} \neg \text{pin_ok} \wedge \mathbf{F} \text{give_money})$

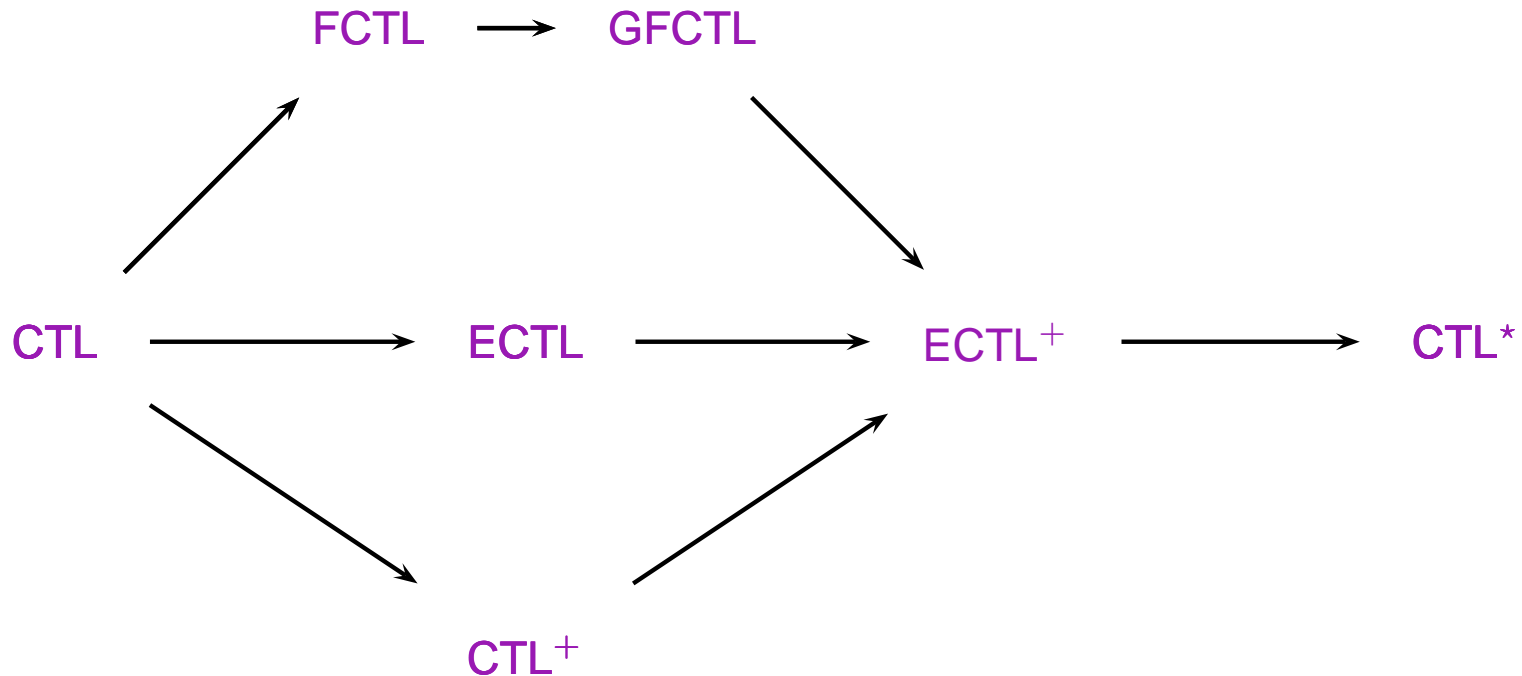
Branching-time temporal logics



FCTL, GFCTL [EL87]: Add fairness conditions to path quantifiers
Strictly more expressive than CTL
Model checking is NP-hard

Example: $\mathbf{A} \tilde{\mathbf{F}}_{\text{wrong_pin}} \wedge \tilde{\mathbf{F}}_{\text{wait_card}} (\mathbf{F} \text{give_money})$

Branching-time temporal logics



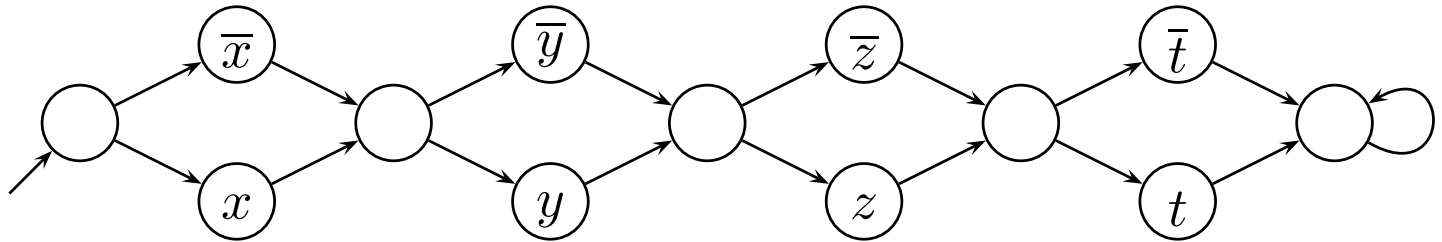
ECTL⁺ [EH86]: Combines **ECTL** and **CTL⁺** extensions
Strictly more expressive than **ECTL** and **CTL⁺**
Model checking is **NP-hard**

Example: $\mathbf{A}(\mathbf{F}^{\infty} \text{wrong_pin} \Rightarrow \mathbf{F} \text{card_back})$

Model checking CTL^+

- Model checking CTL^+ is NP-hard:

SAT : is $(x \vee y \vee z) \wedge (\bar{x} \vee t \vee \bar{z}) \wedge (\bar{x} \vee \bar{t} \vee \bar{y})$ satisfiable ?

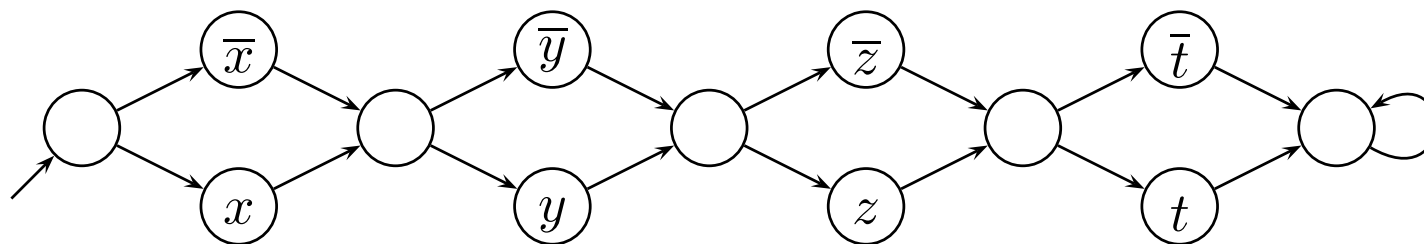


$$\Phi = \mathbf{E}((\mathbf{F} x \vee \mathbf{F} y \vee \mathbf{F} z) \wedge (\mathbf{F} \bar{x} \vee \mathbf{F} t \vee \mathbf{F} \bar{z}) \wedge (\mathbf{F} \bar{x} \vee \mathbf{F} \bar{t} \vee \mathbf{F} \bar{y}))$$

Model checking CTL^+

- Model checking CTL^+ is NP-hard:

SAT : is $(x \vee y \vee z) \wedge (\bar{x} \vee t \vee \bar{z}) \wedge (\bar{x} \vee \bar{t} \vee \bar{y})$ satisfiable ?



$$\Phi = \mathbf{E}((\mathbf{F} x \vee \mathbf{F} y \vee \mathbf{F} z) \wedge (\mathbf{F} \bar{x} \vee \mathbf{F} t \vee \mathbf{F} \bar{z}) \wedge (\mathbf{F} \bar{x} \vee \mathbf{F} \bar{t} \vee \mathbf{F} \bar{y}))$$

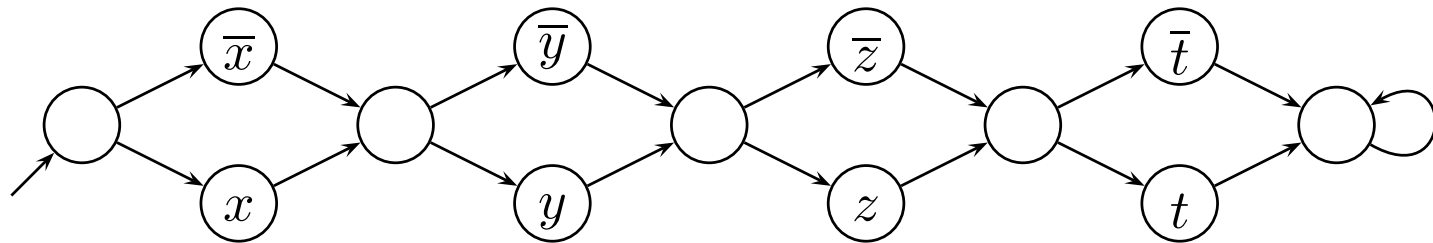
- Model checking a formula $\mathbf{E}\varphi \in \text{CTL}^+$, where φ has no path quantifier, can be done in NP.

Model checking CTL^+ is in $\Delta_2^P = \text{P}^{\text{NP}}$.

Model checking CTL^+

- Model checking CTL^+ is NP-hard:

SAT : is $(x \vee y \vee z) \wedge (\bar{x} \vee t \vee \bar{z}) \wedge (\bar{x} \vee \bar{t} \vee \bar{y})$ satisfiable ?



$$\Phi = \mathbf{E}((\mathbf{F} x \vee \mathbf{F} y \vee \mathbf{F} z) \wedge (\mathbf{F} \bar{x} \vee \mathbf{F} t \vee \mathbf{F} \bar{z}) \wedge (\mathbf{F} \bar{x} \vee \mathbf{F} \bar{t} \vee \mathbf{F} \bar{y}))$$

- Model checking a formula $\mathbf{E}\varphi \in \text{CTL}^+$, where φ has no path quantifier, can be done in NP.

Model checking CTL^+ is in $\Delta_2^P = \text{P}^{\text{NP}}$.

Theorem [LMS01]: Model checking CTL^+ , ECTL^+ and FCTL is Δ_2^P -complete.

The SNSAT problem

Input:

$$\mathcal{I} = \left[\begin{array}{l} x_1 := \exists Z_1 F_1(Z_1), \\ x_2 := \exists Z_2 F_2(x_1, Z_2), \\ \vdots \\ x_n := \exists Z_n F_n(x_1, \dots, x_{n-1}, Z_n) \end{array} \right]$$

\mathcal{I} defines a unique valuation $v_{\mathcal{I}}$ of the variables in X where:

$$v_{\mathcal{I}}(x_i) = \top \text{ iff } F_i(v_{\mathcal{I}}(x_1), \dots, v_{\mathcal{I}}(x_{i-1}), Z_i) \text{ is satisfiable.}$$

Output: Does $v_{\mathcal{I}}(x_n) = \top$?

The SNSAT problem

Input:

$$\mathcal{I} = \left[\begin{array}{l} x_1 := \exists Z_1 F_1(Z_1), \\ x_2 := \exists Z_2 F_2(x_1, Z_2), \\ \vdots \\ x_n := \exists Z_n F_n(x_1, \dots, x_{n-1}, Z_n) \end{array} \right]$$

\mathcal{I} defines a unique valuation $v_{\mathcal{I}}$ of the variables in X where:

$$v_{\mathcal{I}}(x_i) = \top \text{ iff } F_i(v_{\mathcal{I}}(x_1), \dots, v_{\mathcal{I}}(x_{i-1}), Z_i) \text{ is satisfiable.}$$

Output: Does $v_{\mathcal{I}}(x_n) = \top$?

Theorem [LMS01]: SNSAT is Δ_2^P -complete.

Complexity of CTL^+ , ECTL^+ , FCTL

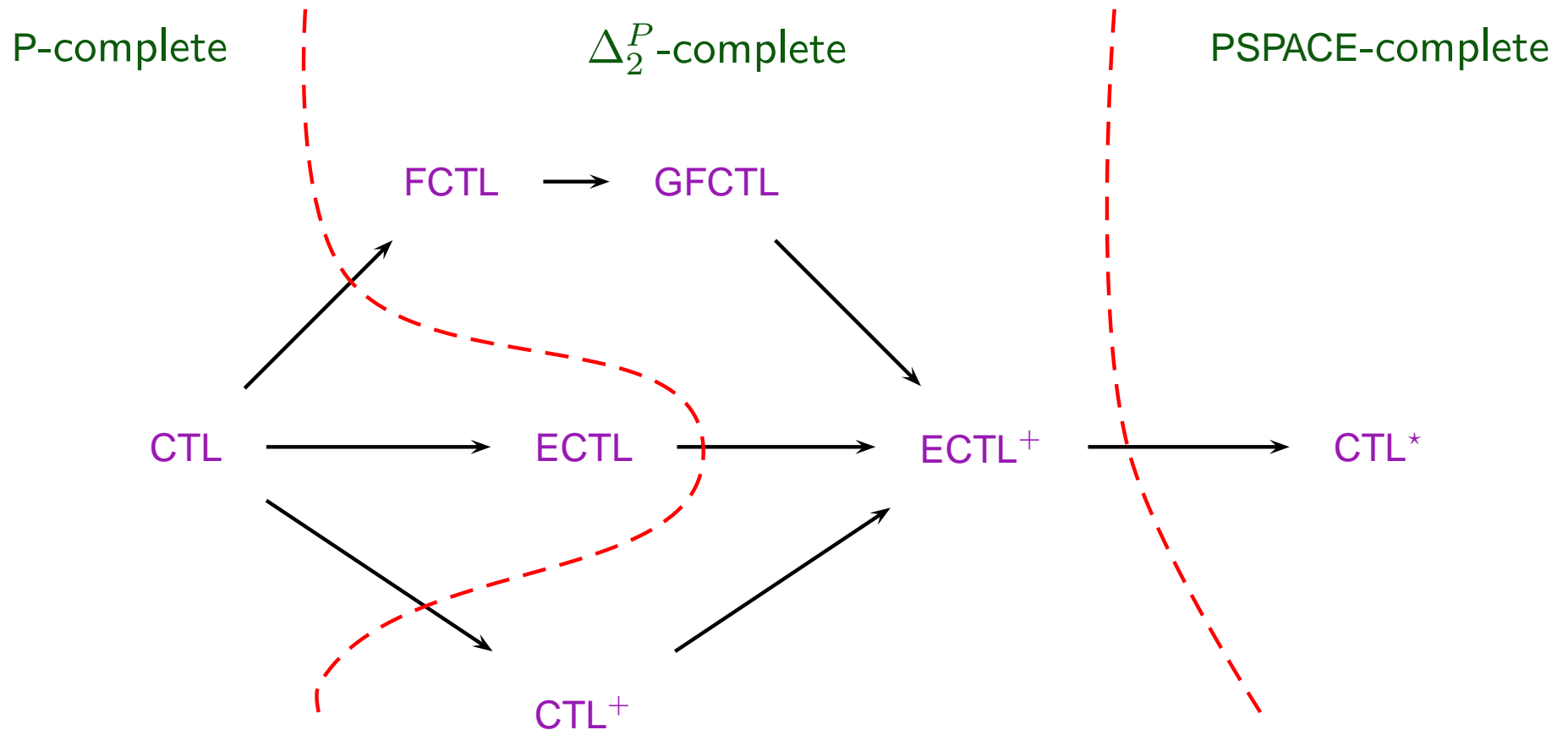
Δ_2^P -hardness for CTL^+ :

Reduction from SNSAT : we build a Kripke structure in which a path represent a valuation of the variables, and a CTL^+ formula expressing that a variable x_i is true iff there is a witness that F_i is satisfiable.

Complexity of CTL^+ , ECTL^+ , FCTL

Δ_2^P -hardness for CTL^+ :

Reduction from **SNSAT**: we build a Kripke structure in which a path represent a valuation of the variables, and a CTL^+ formula expressing that a variable x_i is true iff there is a witness that F_i is satisfiable.



Outline of the talk

1. Past-time modalities in LTL

We prove that past-time modalities *do* add succinctness to LTL, and that they *really* don't change the complexity of model checking.

2. Extensions of CTL

We give optimal algorithms for model checking CTL^+ , FCTL, GFCTL and ECTL^+ : These problems are Δ_2^P -complete.

3. Quantitative temporal logics

We show that it is possible, in certain restricted cases, to perform *timed* model checking in **polynomial time**. We also study several other cases.

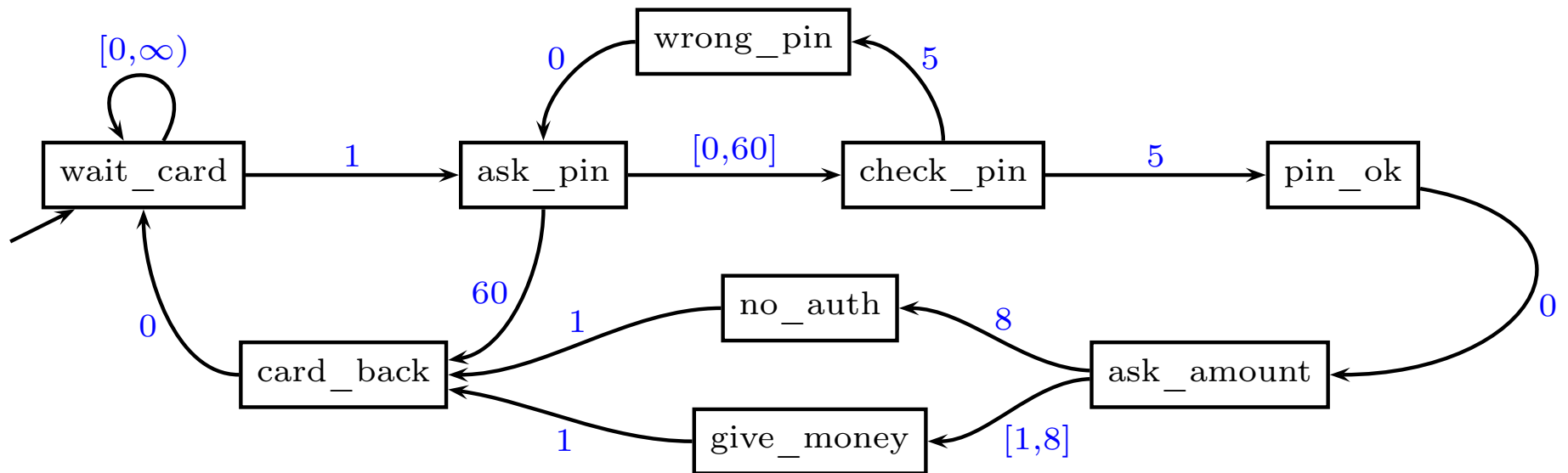
Conclusion

DKS and timed temporal logics

A **Durational Kripke Structure** is a **Kripke structure** whose transitions are labelled with a **duration**.

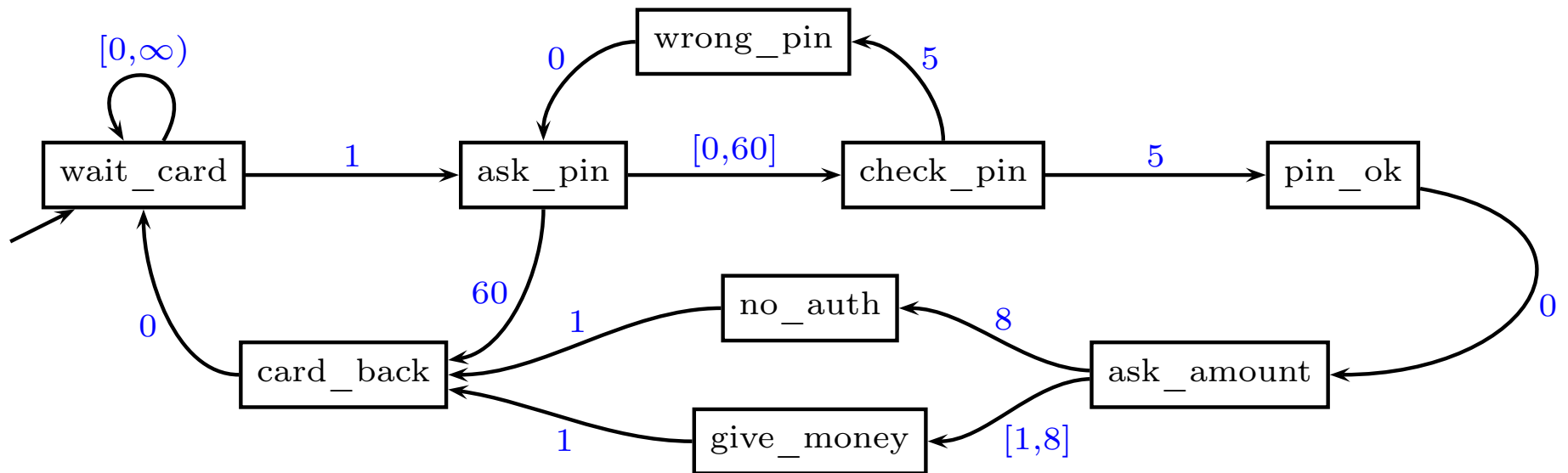
DKS and timed temporal logics

A **Durational Kripke Structure** is a Kripke structure whose transitions are labelled with a **duration**.



DKS and timed temporal logics

A **Durational Kripke Structure** is a Kripke structure whose transitions are labelled with a **duration**.



Example of timed temporal formulas:

AG (pin_ok \Rightarrow **EF**₌₈ card_back)

AG (**EF**_{≤60} wait_card)

With or without exact durations

		ssDKS($\xrightarrow{0/1}$)	tight DKS, DKS
TCTL	\leq, \geq	P-complete	P-complete
	$\leq, =, \geq$		Δ_2^P -complete
TLTL	\leq, \geq	PSPACE-complete	
	$\leq, =, \geq$	EXSPACE-complete	
TCTL*	\leq, \geq	PSPACE-complete	
	$\leq, =, \geq$	EXSPACE-complete	
TCTL ⁺	\leq, \geq	Δ_2^P -complete	
	$\leq, =, \geq$		

Outline of the talk

1. Past-time modalities in LTL

We prove that past-time modalities *do* add succinctness to LTL, and that they *really* don't change the complexity of model checking.

2. Extensions of CTL

We give optimal algorithms for model checking CTL^+ , FCTL, GFCTL and ECTL^+ : These problems are Δ_2^P -complete.

3. Quantitative temporal logics

We show that it is possible, in certain restricted cases, to perform *timed* model checking in polynomial time. We also study several other cases.

Conclusion

Conclusion

We solved several problems related to the expressiveness and complexity of various different logics.

- **LTL** should be extended with past modalities, since they make specification easier (more succinct and more natural), and are not harder to verify.

The **N** operator also brings succinctness, but verification becomes harder.

Conclusion

We solved several problems related to the expressiveness and complexity of various different logics.

- **LTL** should be extended with past modalities, since they make specification easier (more succinct and more natural), and are not harder to verify.

The **N** operator also brings succinctness, but verification becomes harder.

- In **CTL**, allowing the boolean combination of temporal statements (possibly fairness) in the scope of path quantifiers makes model checking much harder.

These were the first verification problems known to be complete for Δ_2^P .

Conclusion

We solved several problems related to the expressiveness and complexity of various different logics.

- **LTL** should be extended with past modalities, since they make specification easier (more succinct and more natural), and are not harder to verify.

The **N** operator also brings succinctness, but verification becomes harder.

- In **CTL**, allowing the boolean combination of temporal statements (possibly fairness) in the scope of path quantifiers makes model checking much harder.

These were the first verification problems known to be complete for Δ_2^P .

- It is possible to perform timed model checking in **polynomial time**.

Model checking timed properties is harder when allowing exact constraints.

Future work

- still many open questions concerning **expressiveness** and **complexity** of temporal logics
- implementation of past modalities into **LTL** model checkers,
- model checking a single path,
- study different semantics for durations in DKS.