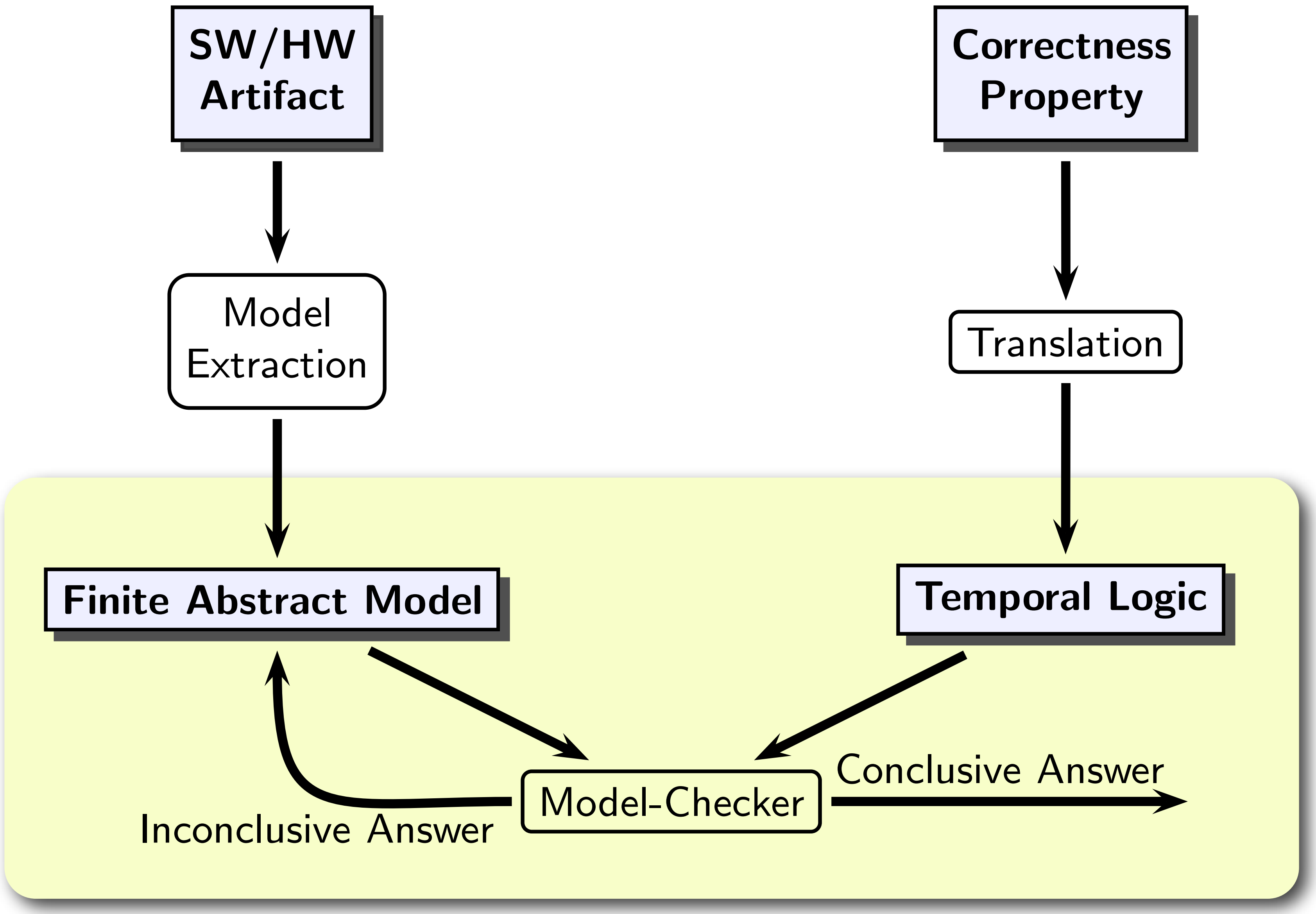
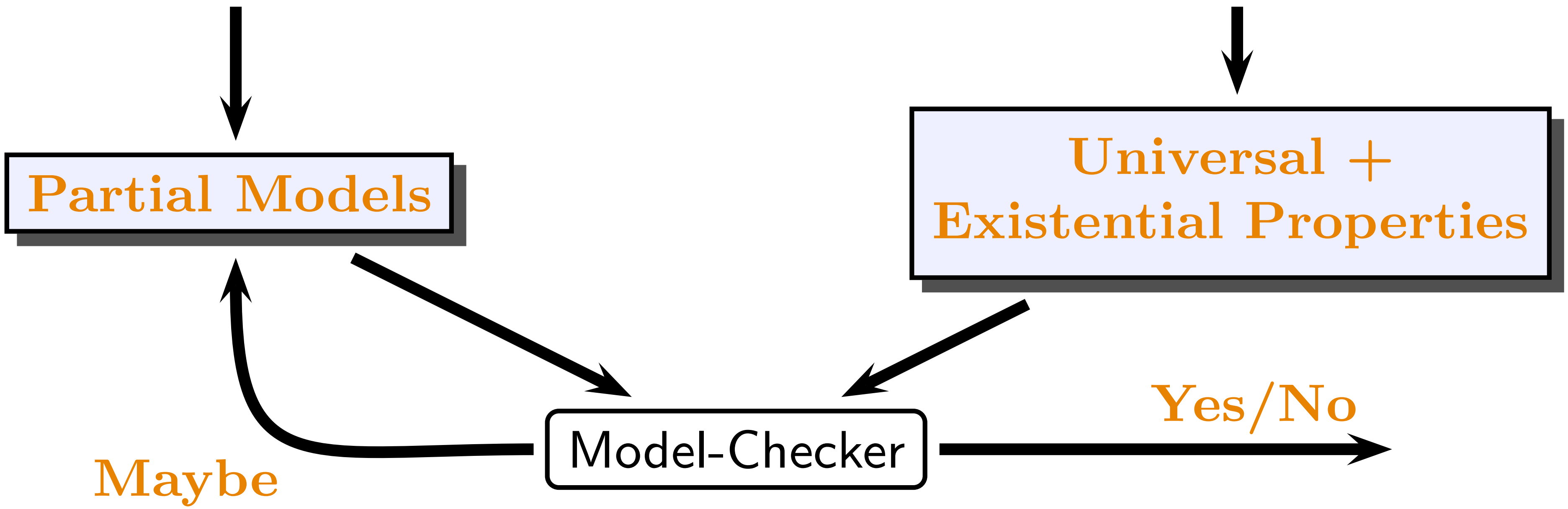
Thorough Checking Revisited

Shiva Nejati Mihaela Gheorghiu Marsha Chechik
{shiva,mg,chechik}@cs.toronto.edu
University of Toronto

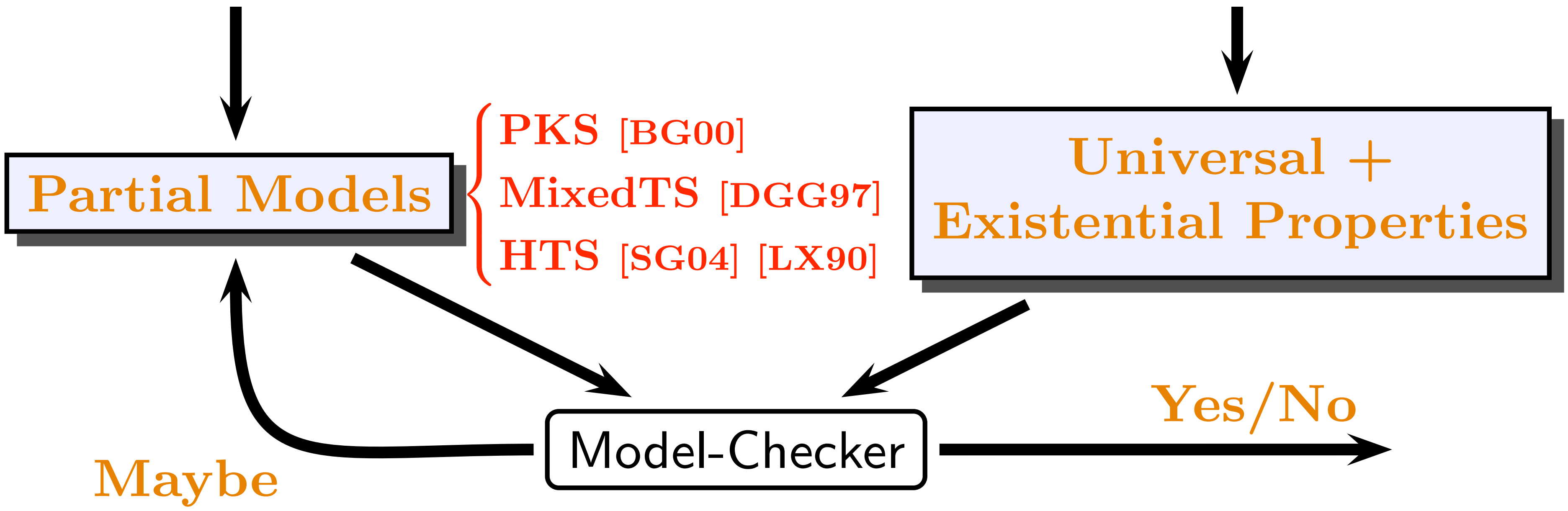
Automated Abstraction



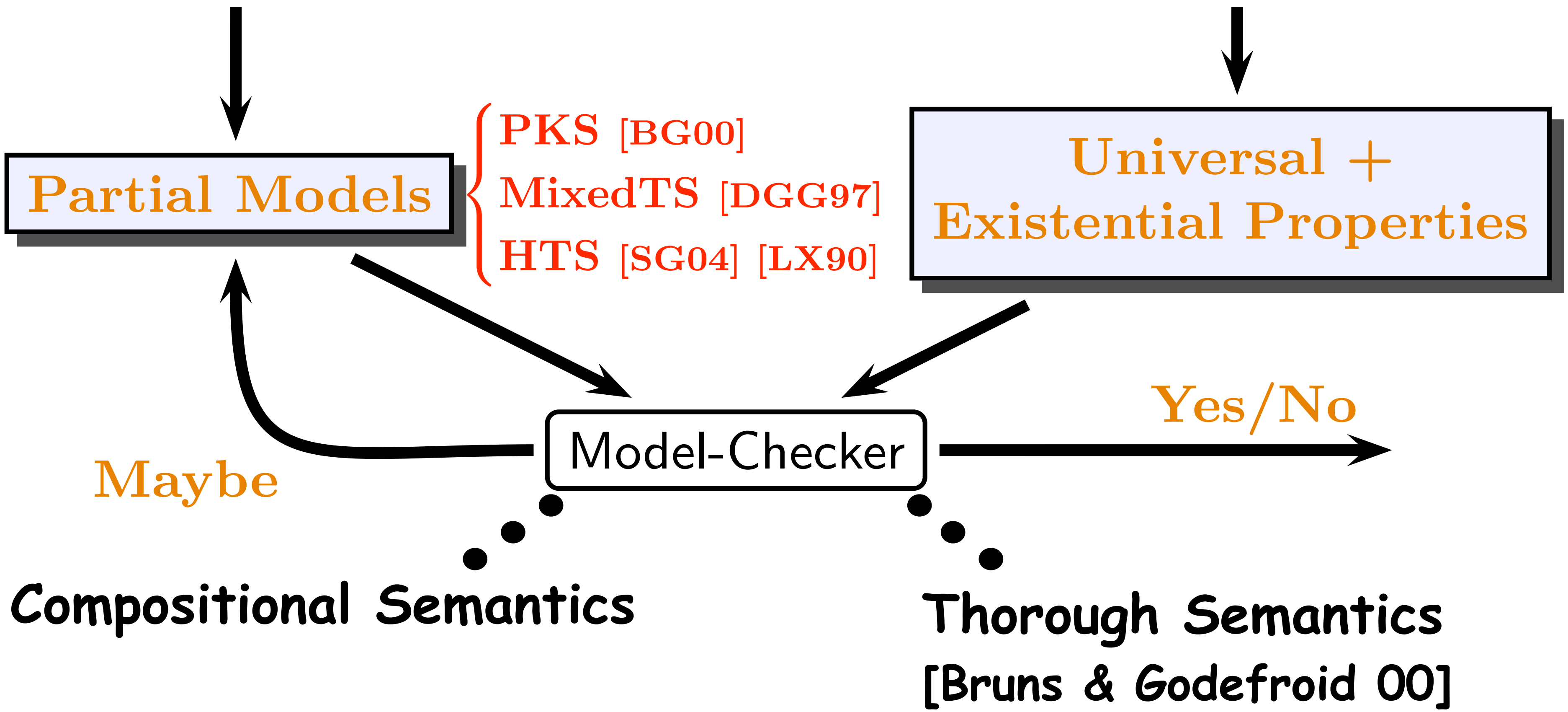
3-Valued Abstraction



3-Valued Abstraction



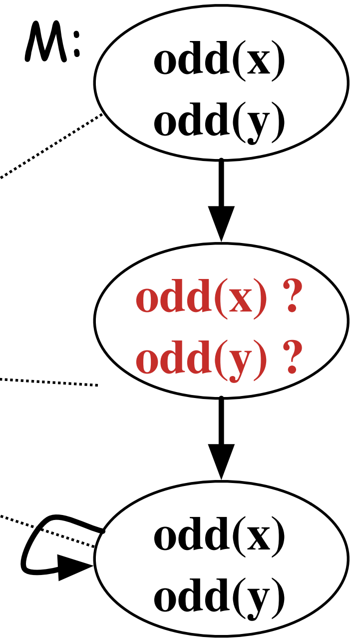
3-Valued Abstraction



3-Valued Semantics: Example

P:

```
int x, y = 1, 1;
int t;
x, y = t, t+1;
x, y = 1, 1;
```

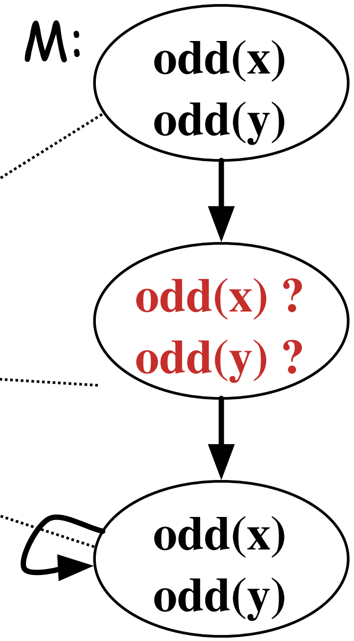


Property : $AG(odd(y)) \wedge A[odd(x)\ U\ \neg odd(y)]$

Compositional Semantics	
Thorough Semantics	

3-Valued Semantics: Example

M:



P:

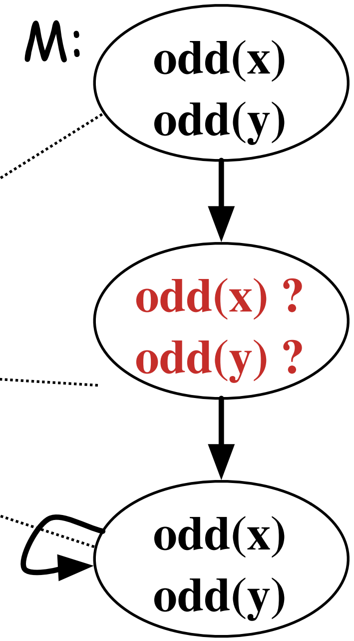
```
 int x, y = 1, 1;
 int t;
 x, y = t, t+1;
 x, y = 1, 1;
```

Property : $AG(odd(y)) \wedge A[odd(x)\ U\ \neg odd(y)]$

Compositional Semantics	$AG(odd(y)) \wedge A[odd(x)\ U\ \neg odd(y)]$
Thorough Semantics	

3-Valued Semantics: Example

P:
```
int x, y = 1, 1;
int t;
x, y = t, t+1;
x, y = 1, 1;
```

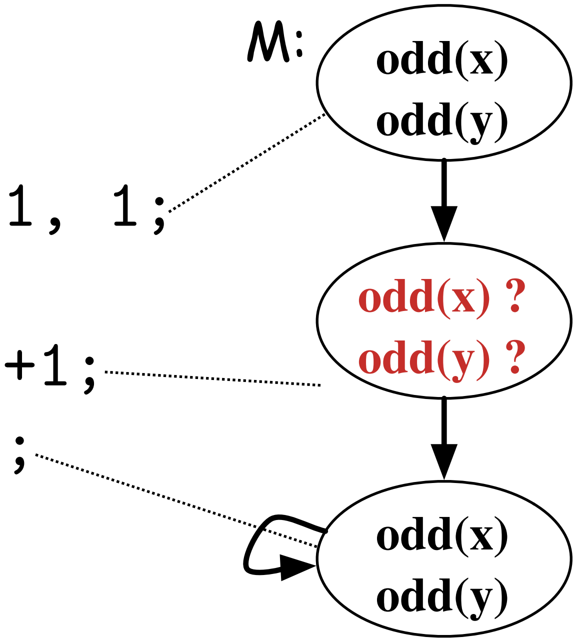


Property : AG(odd(y)) ∧ A[odd(x) U ¬odd(y)]

Compositional Semantics	Maybe ∧ A[odd(x) U ¬odd(y)]
Thorough Semantics	

3-Valued Semantics: Example

```
P:
 int x, y = 1, 1;
 int t;
 x, y = t, t+1;
 x, y = 1, 1;
```



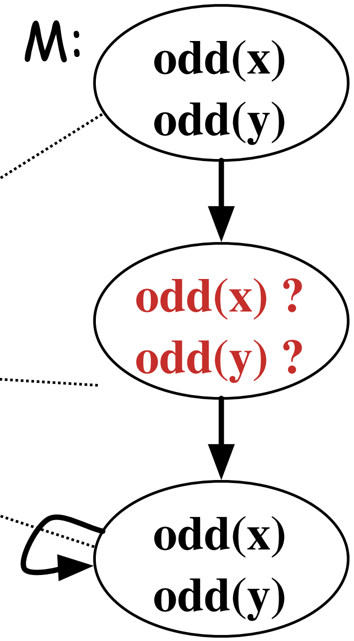
Property : $AG(odd(y)) \wedge A[odd(x)\ U\ \neg odd(y)]$

Compositional Semantics	Maybe $\wedge$ Maybe
Thorough Semantics	

3-Valued Semantics: Example

P:

```
int x, y = 1, 1;
int t;
x, y = t, t+1;
x, y = 1, 1;
```



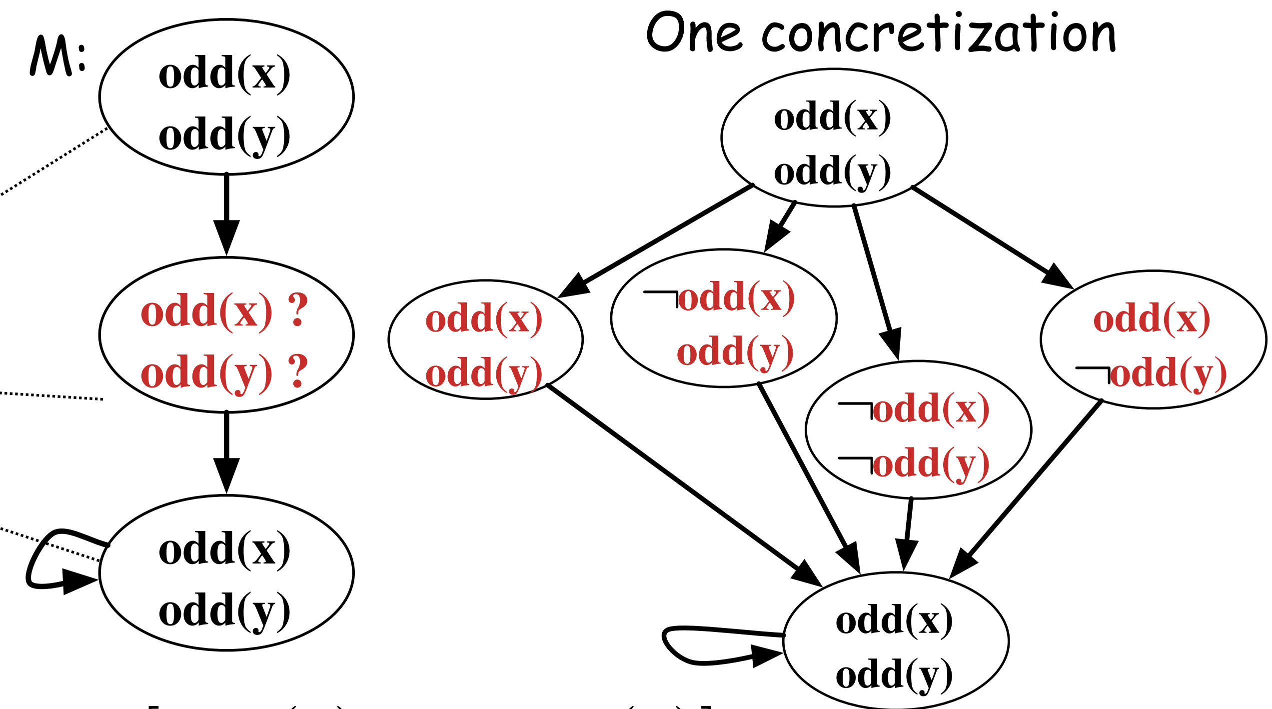
Property : $AG(odd(y)) \wedge A[odd(x)\ U\ \neg odd(y)]$

Compositional Semantics	Maybe
Thorough Semantics	

3-Valued Semantics: Example

P:

```
 int x, y = 1, 1;
 int t;
 x, y = t, t+1;
 x, y = 1, 1;
```



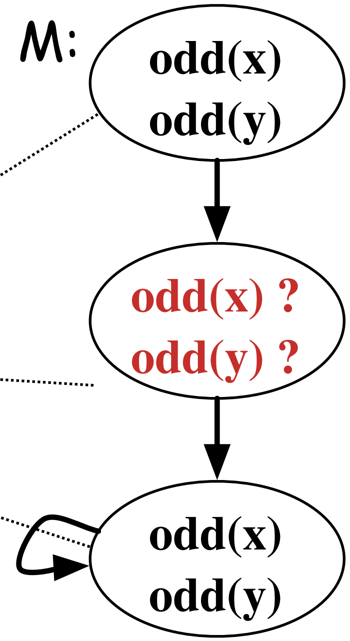
Property : $AG(odd(y)) \wedge A[odd(x)\ U\ \neg odd(y)]$

Compositional Semantics	Maybe
Thorough Semantics	$AG(odd(y)) \wedge A[odd(x)\ U\ \neg odd(y)]$ False over all Concretizations of M

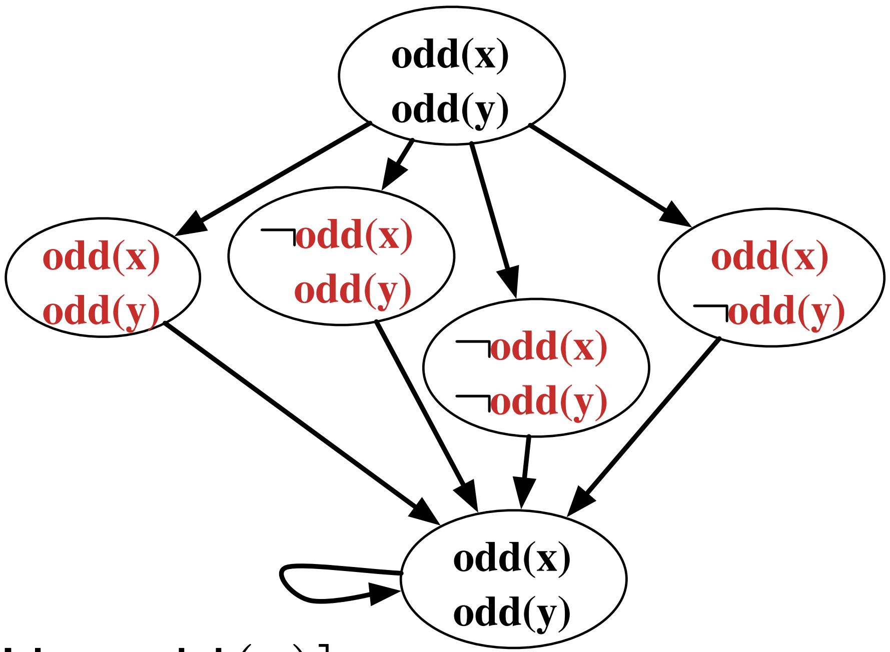
3-Valued Semantics: Example

```
P:
 int x, y = 1, 1;
 int t;
 x, y = t, t+1;
 x, y = 1, 1;
```

M:



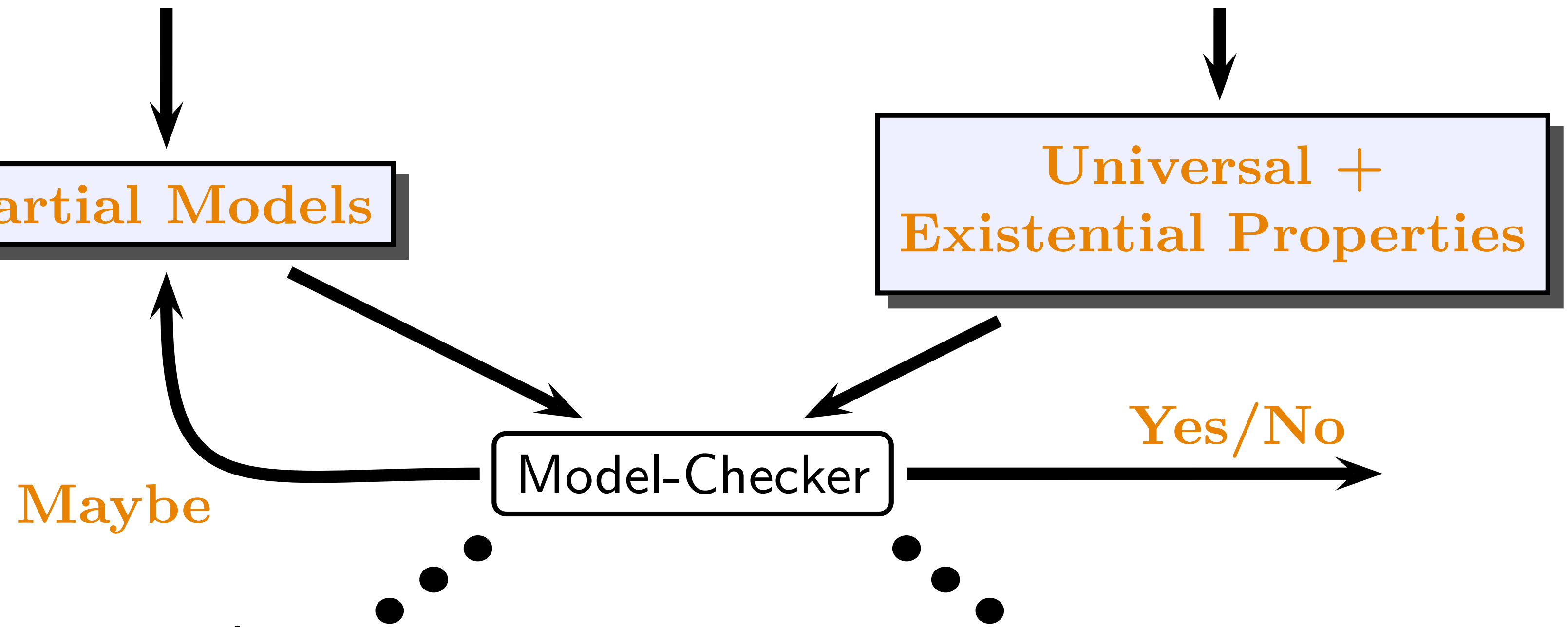
One concretization



Property : $AG(odd(y)) \wedge A[odd(x)\ U\ \neg odd(y)]$

Compositional Semantics	Maybe
Thorough Semantics	False

Compositional vs Thorough



Compositional Semantics

- ✔ Computationally cheap
- ✘ Less precise (more maybe's)
- ✔ Various implementations

Thorough Semantics

- ✘ Computationally expensive
- ✔ More precise (less maybe's)
- ✘ No implementation

Need to increase conclusiveness while avoiding too much overhead

Implementing Thorough via Compositional

- Identify formulas where compositional = thorough
  - Self-minimizing formulas [Godefroid & Huth 05]
  - E.g. $AG(odd(y))$
- Transform other formulas into equivalent self-minimizing ones
  - Semantic minimization [Reps et. al. 02]
  - E.g. $AG(odd(y)) \wedge A[odd(x)\ U\ \neg odd(y)] \triangleq$

    $A[(odd(x) \wedge odd(y))\ U\ False]$ (Self-minimizing)

Thorough Checking Algorithm

ThoroughCheck(M, φ)
(1): if $(v := \textsc{ModelCheck}(M, \varphi)) \neq$ `Maybe`
  return v
(2): if $\textsc{IsSelfMinimizing}(M, \varphi)$
  return `Maybe`
(3): return $\textsc{ModelCheck}(M, \textsc{SemanticMinimization}(\varphi))$

Thorough Checking Algorithm

$$
\begin{array}{ll}
\textbf{ThoroughCheck}(M, \varphi) & \\
(1): & \text{if } (v := \textsc{ModelCheck}(M, \varphi)) \neq \texttt{Maybe} ✔ \\
 & \quad \text{return } v \\
(2): & \text{if } \textsc{IsSelfMinimizing}(M, \varphi) \\
 & \quad \text{return } \texttt{Maybe} \\
(3): & \text{return } \textsc{ModelCheck}(M, \textsc{SemanticMinimization}(\varphi))
\end{array}
$$

Our Goal

ThoroughCheck(M, φ)

(1): if $(v := \textsc{ModelCheck}(M, \varphi)) \neq$ `Maybe` ✔
 return v

(2): if $\textsc{IsSelfMinimizing}(M, \varphi)$
 return `Maybe`

(3): return $\textsc{ModelCheck}(M, \textsc{SemanticMinimization}(\varphi))$

- **Step (2):**
  - Identifying a large class of self-minimizing formulas
- **Step (3):**
  - Devising practical algorithms for semantic minimization of remaining formulas

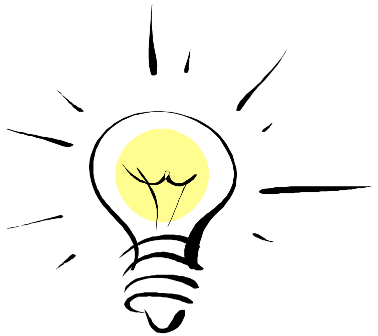
Our Contributions

1. We prove that disjunctive/conjunctive μ-calculus formulas are self-minimizing
   - Related Work:
     - [Gurfinkel & Chechik 05] [Godefroid & Huth 05] checking pure polarity
     - Only works for PKSs, not for all partial models
2. We provide a semantic minimization algorithm via the tableau-based translation of [Janin & Walukiewicz 95]
   - Related Work:
     - [Godefroid & Huth 05]: μ-calculus is closed under semantic-minimization
     - But no implementable algorithm

Main Idea

- Thorough checking can be as hard as satisfiability checking
  - Satisfiability checking is linear for disjunctive μ-calculus
    - Then, can we show that disjunctive μ-calculus is self-minimizing?
    - But, a naive inductive proof does not work for the greatest fixpoint formulas [Godefroid & Huth 05]

- Our proof uses an automata characterization of thorough checking
  - reducing checking self-minimization to deciding an automata intersection game



Outline

- Need for thorough checking
- Thorough via compositional
- Main Result: Disjunctive/Conjunctive μ-calculus is self-minimizing
  - Intuition
  - Background
  - Proof
- Our thorough checking algorithm
- Conclusion and future work

Background

- Disjunctive μ-calculus [Janin and Walukiewicz 95]
  - Conjunctions are restricted (special conjunctions)
  - Examples

$$\varphi_1 = \mathsf{EXp} \wedge \mathsf{EX}\neg\mathsf{q} \wedge \mathsf{AX}(\mathsf{p} \vee \neg\mathsf{q}) \quad ✔$$

$$\varphi_2 = \mathsf{AX}(\mathsf{p} \wedge \mathsf{q}) \quad ✔$$

$$\varphi_3 = \mathsf{AXp} \wedge \mathsf{AXq} \quad ✘$$

  - Syntax

$$\varphi ::= \mathsf{p} \mid \neg\mathsf{p} \mid \mathsf{Z} \mid \varphi \vee \varphi \mid \mathbf{p} \wedge \bigwedge_{\psi\in\Gamma} \mathbf{EX}\psi \wedge \mathbf{AX} \bigvee_{\psi\in\Gamma} \psi \mid \nu(\mathsf{Z})\cdot\varphi(\mathsf{Z}) \mid \mu(\mathsf{Z})\cdot\varphi(\mathsf{Z})$$

- Conjunctive μ-calculus is dual
- Disjunctive μ-calculus is equal to μ-calculus

Background: Abstraction as Automata [Dams & Namjoshi 05]

- Formulas = automata, abstract models = automata
  - Model Checking
    - Model M satisfies formula φ $\mathcal{L}(\mathcal{A}_{\mathsf{M}}) \subseteq \mathcal{L}(\mathcal{A}_{\varphi})$
  - Refinement Checking
    - Model M abstracts model M' $\mathcal{L}(\mathcal{A}_{\mathsf{M}}) \subseteq \mathcal{L}(\mathcal{A}_{\mathsf{M}'})$
- We use μ-automata [Janin & Walukiewicz 95]
  - Similar to non-deterministic tree automata
  - But
    - no fixed branching degree
    - no ordering over successors

Self-minimization and Automata

- A formula φ is self-minimizing if
  1. For every abstract model M over which φ is non-false (true or maybe)

     there is a completion of M satisfying φ
  2. For every abstract model M over which φ is non-true (false or maybe)

     there is a completion of M refuting φ

Self-minimization and Automata

- A formula φ is self-minimizing if
  1. For every abstract model M over which φ is non-false (true or maybe)

     $$\mathcal{L}(\mathcal{A}_{\mathsf{M}}) \cap \mathcal{L}(\mathcal{A}_{\varphi}) \neq \emptyset$$

  2. For every abstract model M over which φ is non-true (false or maybe)

     there is a completion of M refuting φ

Self-minimization and Automata

- A formula φ is self-minimizing if
  1. For every abstract model M over which φ is non-false (true or maybe)

$$\mathcal{L}(\mathcal{A}_{\mathsf{M}}) \cap \mathcal{L}(\mathcal{A}_{\varphi}) \neq \emptyset$$

  2. For every abstract model M over which φ is non-true (false or maybe)

$$\mathcal{L}(\mathcal{A}_{\mathsf{M}}) \cap \mathcal{L}(\mathcal{A}_{\neg\varphi}) \neq \emptyset$$

Self-minimization and Automata

- A formula φ is self-minimizing if
  1. For every abstract model M over which φ is non-false (true or maybe)

     $$\mathcal{L}(\mathcal{A}_{\mathsf{M}}) \cap \mathcal{L}(\mathcal{A}_{\varphi}) \neq \emptyset$$

  2. For every abstract model M over which φ is non-true (false or maybe)

     $$\mathcal{L}(\mathcal{A}_{\mathsf{M}}) \cap \mathcal{L}(\mathcal{A}_{\neg\varphi}) \neq \emptyset$$

- Existing partial model formalisms can be translated to μ-automata
- There exists a linear syntactic translation from disjunctive μ-calculus to μ-automata [Janin & Walukiewicz 95]

Outline

- Need for thorough checking
- Thorough via compositional
- Main Result: Disjunctive/Conjunctive μ-calculus is self-minimizing
  - Intuition
  - Background
  - Proof
- Our thorough checking algorithm
- Conclusion and future work

Main Result

→ Let φ be a disjunctive formula. Show:

for every abstract model M over which φ is non-false

$$\mathcal{L}(\mathcal{A}_{\mathsf{M}}) \cap \mathcal{L}(\mathcal{A}_{\varphi}) \neq \emptyset$$

→ The case for conjunctive φ is dual

→ Proof Steps:

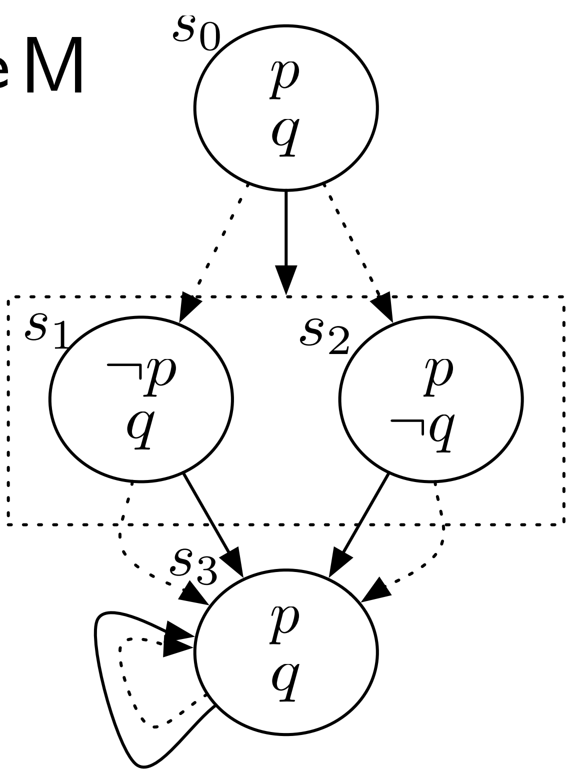
1. Translate models and formulas to μ-automata
2. Find a winning strategy for an intersection game between $\mathcal{A}_{\mathsf{M}}$ and $\mathcal{A}_{\varphi}$ (by structural induction)

Illustrating the Proof

- **Show that AGp is self-minimizing**
  - i.e., ∀M over which φ is non-false

$$\mathcal{L}(\mathcal{A}_{\mathsf{M}}) \cap \mathcal{L}(\mathcal{A}_{\mathsf{AGP}}) \neq \emptyset$$

Choose M



AGp

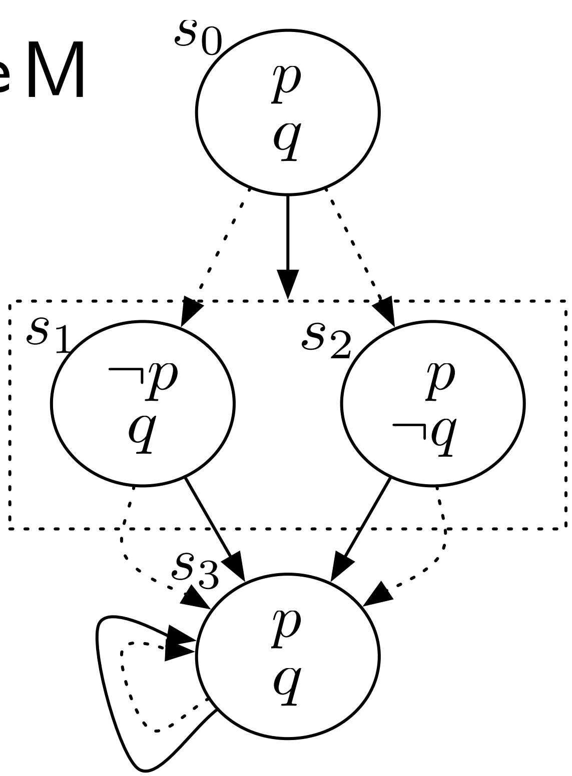
Illustrating the Proof

- **Show that AGp is self-minimizing**
  - i.e., ∀M over which φ is non-false

$$\mathcal{L}(\mathcal{A}_{\mathsf{M}}) \cap \mathcal{L}(\mathcal{A}_{\mathsf{AGP}}) \neq \emptyset$$

1. Translate models and formulas to μ-automata

Choose M



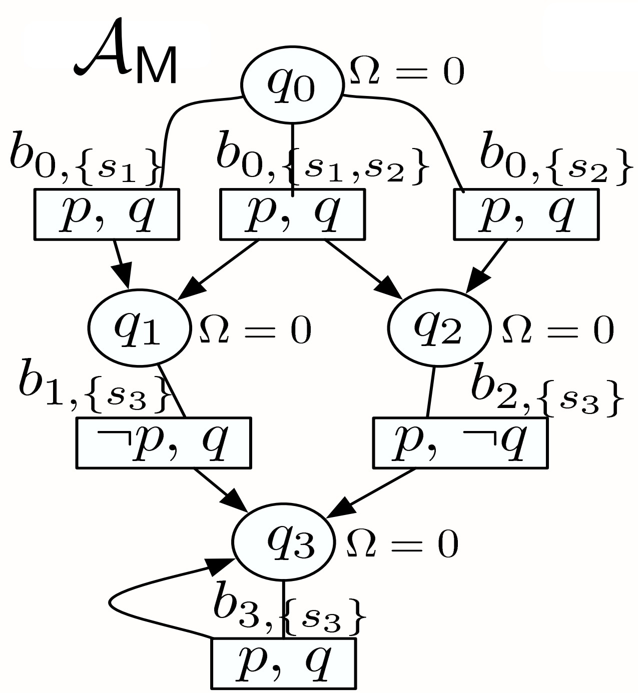
AGp

Illustrating the Proof

- **Show that AGp is self-minimizing**
  - **i.e., ∀M over which φ is non-false**

$$\mathcal{L}(\mathcal{A}_{\mathsf{M}}) \cap \mathcal{L}(\mathcal{A}_{\mathsf{AGP}}) \neq \emptyset$$

1. Translate models and formulas to μ-automata



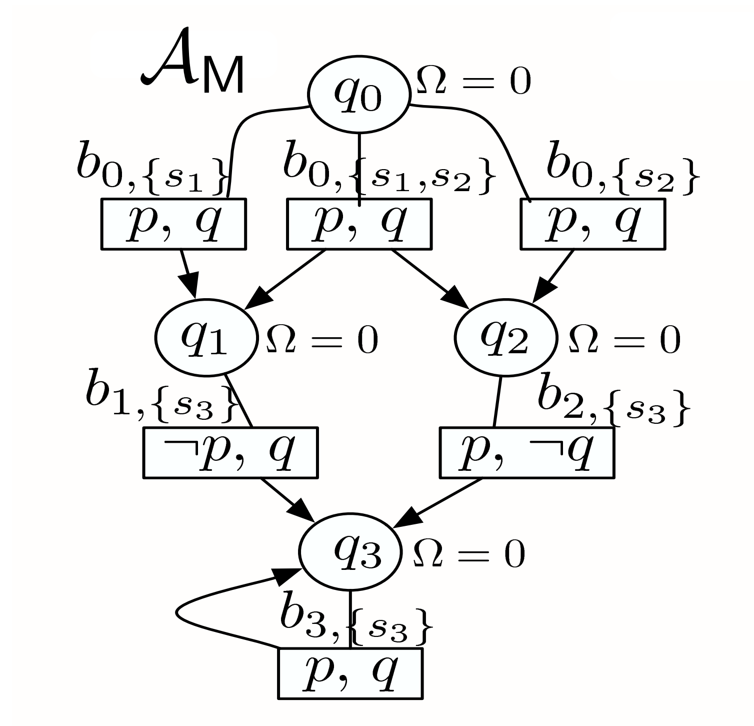
AGp

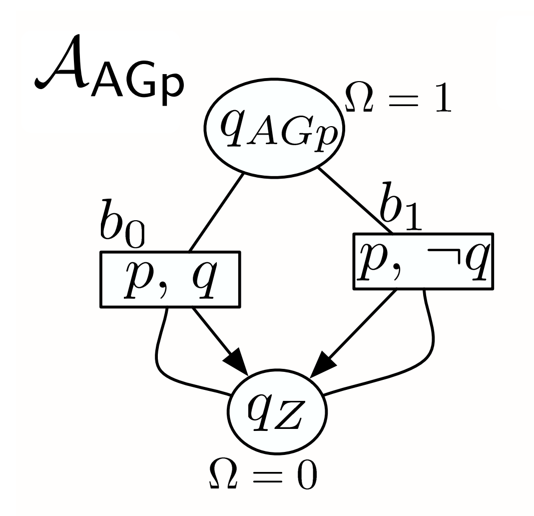
Illustrating the Proof

- Show that AGp is self-minimizing
  - i.e., ∀M over which φ is non-false

$$\mathcal{L}(\mathcal{A}_{\mathsf{M}}) \cap \mathcal{L}(\mathcal{A}_{\mathsf{AGP}}) \neq \emptyset$$

1. Translate models and formulas to μ-automata



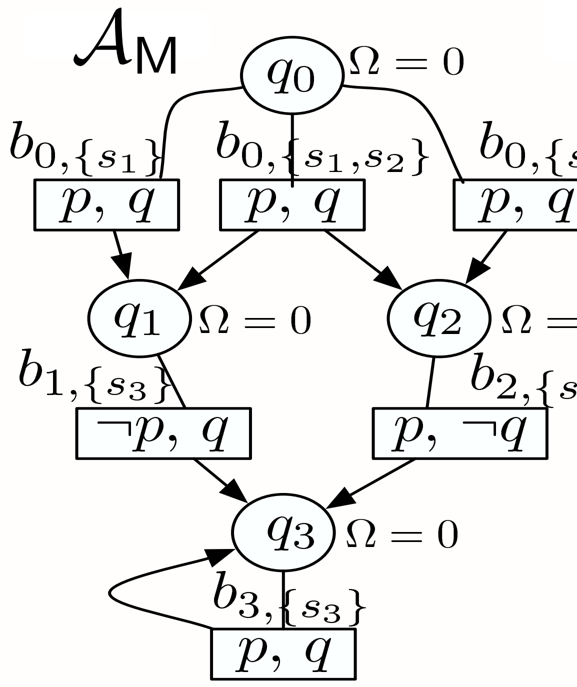


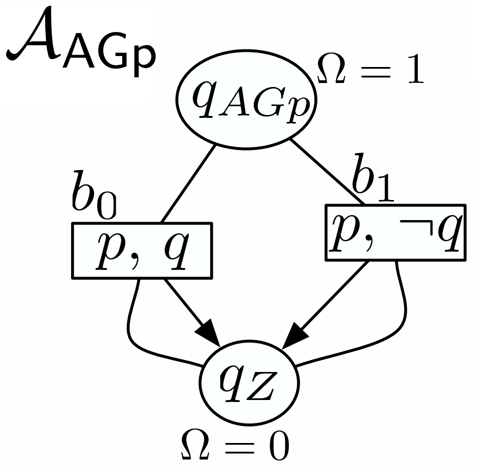
Illustrating the Proof

- Show that AGp is self-minimizing
  - i.e., ∀M over which φ is non-false

$$\mathcal{L}(\mathcal{A}_{\mathsf{M}}) \cap \mathcal{L}(\mathcal{A}_{\mathsf{AGP}}) \neq \emptyset$$

2. Find a winning strategy for an intersection game



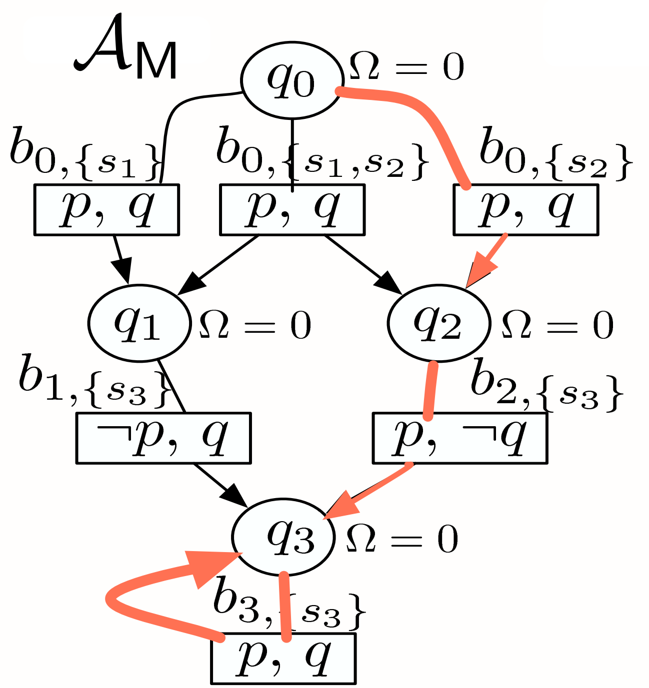


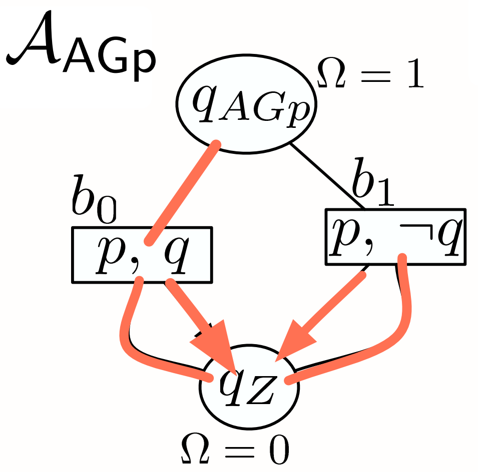
Illustrating the Proof

- Show that AGp is self-minimizing
  - i.e., ∀M over which φ is non-false

$$\mathcal{L}(\mathcal{A}_{\mathsf{M}}) \cap \mathcal{L}(\mathcal{A}_{\mathsf{AGP}}) \neq \emptyset$$

2. Find a winning strategy for an intersection game



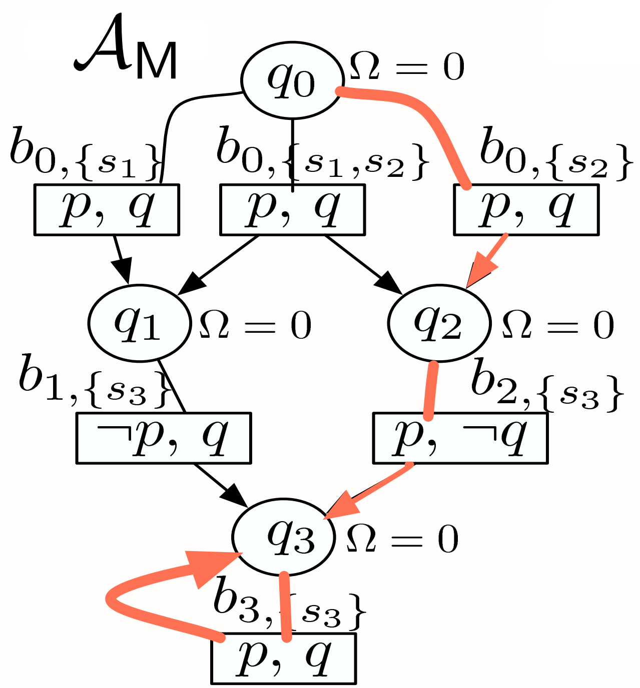


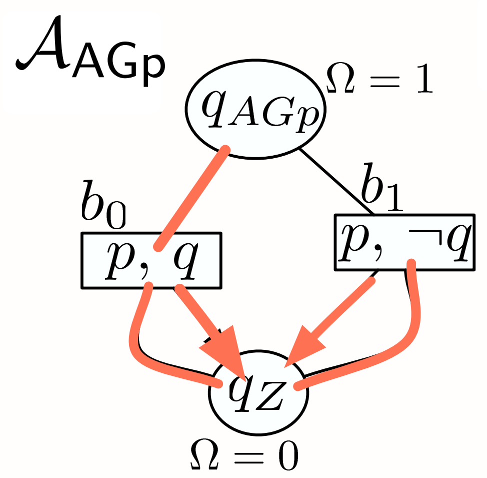
Illustrating the Proof

- Show that AGp is self-minimizing
  - i.e., ∀M over which φ is non-false

$$\mathcal{L}(\mathcal{A}_{\mathsf{M}}) \cap \mathcal{L}(\mathcal{A}_{\mathsf{AGP}}) \neq \emptyset$$

2. Find a winning strategy for an intersection game





Proof by structural induction (see the paper)

Main Result

- Proof Steps:
  1. Translate models and formulas to μ-automata
  2. Find a winning strategy for an intersection game
- In conclusion:
  - Disjunctive/conjunctive μ-calculus formulas are self-minimizing
  - Every μ-calculus formula can be translated to its disjunctive/conjunctive form

Outline

- Need for thorough checking
- Thorough via compositional
- Main Result: Disjunctive/Conjunctive μ-calculus is self-minimizing
  - Intuition
  - Background
  - proof
- Our thorough checking algorithm
- Conclusion and future work

Thorough Checking Algorithm

ThoroughCheck(M, φ)

(1): if $(v := \textsc{ModelCheck}(M, \varphi)) \neq$ `Maybe`

return v

(2): if $\textsc{IsSelfMinimizing}(M, \varphi)$

return `Maybe`

(3): return $\textsc{ModelCheck}(M, \textsc{SemanticMinimization}(\varphi))$

Self-Minimization

IsSelfMinimizing(M, φ)

(i) if M is a PKS or an MixTS and φ is monotone
return `true`

(ii) if M is an HTS and φ is disjunctive
return `true`

(iii) return `false`

- **Example**
  - **Property** $AGq \wedge A[p\ U\ \neg q]$ **over**
    - **PKSs and MixTSs violates condition (i)**
    - **HTSs violates condition (ii)**
  - **Thus,** $AGq \wedge A[p\ U\ \neg q]$ **is not self-minimizing**

Semantic Minimization

SemanticMinimization(φ)

(i) convert φ to its disjunctive form $\varphi^{\vee}$

(ii) replace all special conjunctions in $\varphi^{\vee}$ containing p and $\neg p$ with `False`

(iii) return $\varphi^{\vee}$

→ **Example: semantic minimization of** $\mathsf{AGq} \wedge \mathsf{A[p\ U\ \neg q]}$

- **Step (i)** $\mathsf{AGq} \wedge \mathsf{A[p\ U\ \neg q]} \stackrel{(i)}{\rightarrow} \mathsf{A[p \wedge q\ U\ q \wedge \neg q \wedge AXAGq]}$
- **Step (ii)** $\mathsf{A[p \wedge q\ U\ q \wedge \neg q \wedge AXAGq]} \stackrel{(ii)}{\rightarrow} \mathsf{A[p \wedge q\ U\ False]}$

Complexity

ThoroughCheck(M, φ)

(1): if $(v := \textsc{ModelCheck}(M, \varphi)) \neq$ `Maybe`
    return v

(2): if $\textsc{IsSelfMinimizing}(M, \varphi)$
    return `Maybe`

(3): return $\textsc{ModelCheck}(M, \textsc{SemanticMinimization}(\varphi))$

- **Step (1)**
  - Model checking μ-calculus formulas $O((|\varphi| \cdot |M|)^{\lfloor d/2 \rfloor + 1})$
- **Step (2)**
  - Self-minimization check is linear in the size of formulas
- **Step (3)**
  - Semantic minimization $O((2^{O(|\varphi|)} \cdot |M|)^{\lfloor d/2 \rfloor + 1})$

Conclusion

- Studied thorough checking over partial models
  - An automata-based characterization for thorough checking
  - Simple and syntactic self-minimization checks
    - Grammars for identifying self-minimizing formulas in CTL
  - A semantic-minimization procedure

Future Work

- Studying the classes of formulas for which thorough checking is cheap
  - linear in the size of models
- Identifying commonly used formulas in practice that are self-minimizing

Thank You!

Questions?