On Module Checking and Strategies

Wojtek Jamroga, University of Luxembourg
Aniello Murano, University of Naples

HIGHLIGHTS @ Paris, 5th of September 2014

Outline

1. Introduction
2. Verification of Open Systems
3. Results
4. Conclusions

Outline

1 Introduction

2 Verification of Open Systems

3 Results

4 Conclusions

Motivation

- Model checking is one of the most successful technologies for reasoning about temporal specifications of systems:

$$M \models \varphi$$

- Model checking typically applies to closed systems whose behavior is fully specified.
- Nowadays more and more systems are open, i.e., only a part of the world is under the control of the system and the rest is up to the (uncontrollable) environment.

Verification of Open Systems

- Verification of open systems $\rightsquigarrow$ module checking (1996)

  (Idea: take the system to be a module embedded in an environment, and check the specification for all possible behaviors of the environment)

- Thus, in order to verify an open system, we change the decision problem

Verification of Open Systems

- Verification of open systems $\rightsquigarrow$ module checking (1996)

  (Idea: take the system to be a module embedded in an environment, and check the specification for all possible behaviors of the environment)

- Thus, in order to verify an open system, we change the decision problem

- Alternative: keep the decision problem (= model checking) but change the logic
- Alternating-time logic (ATL) has been proposed in 1997 specifically for specification and verification of open systems

Module Checking vs. Model Checking

- Two verification problems are very close in spirit:
  - $\rightsquigarrow$ module checking of CTL, and
  - $\rightsquigarrow$ model checking of ATL

Module Checking vs. Model Checking

- Two verification problems are very close in spirit:
  - ⇝ module checking of CTL, and
  - ⇝ model checking of ATL
- The latter seems a natural multi-agent extension of the former
- ...and it is commonly believed that model checking of ATL subsumes module checking of CTL in a straightforward way

Module Checking vs. Model Checking

- Two verification problems are very close in spirit:
  - ⇝ module checking of CTL, and
  - ⇝ model checking of ATL
- The latter seems a natural multi-agent extension of the former
- ...and it is commonly believed that model checking of ATL subsumes module checking of CTL in a straightforward way
- However, the exact relationship has never been established

Contribution in a Sentence (or Two)

1. We show that, contrary to popular belief, module checking of CTL is **not** a special case of model checking ATL

Contribution in a Sentence (or Two)

1. We show that, contrary to popular belief, module checking of CTL is **not** a special case of model checking ATL
2. We also show that, in order to embed the former in the latter, a significantly different semantics must be used for ATL

Outline

1 Introduction

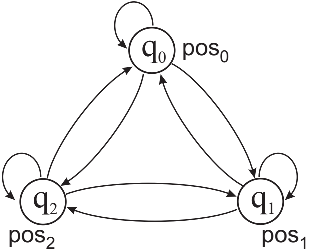
2 Verification of Open Systems

3 Results

4 Conclusions

CTL Model Checking

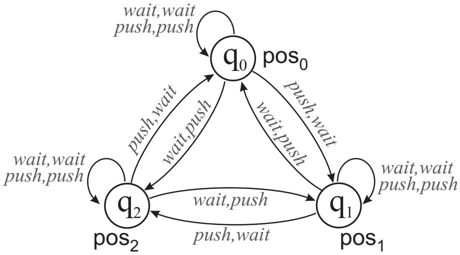
- Given a Kripke structure (KS) M and a CTL formula φ, determine whether $M \models \varphi$



- An example:
  - ♣ A◇pos$_1$ (for all paths, M will eventually reach pos_1)

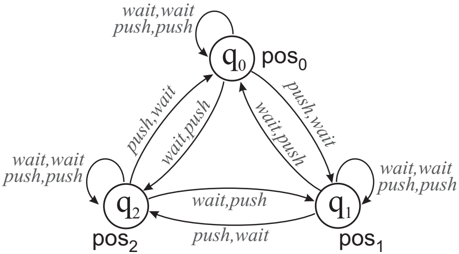
ATL Model Checking

- ATL: temporal logic meets strategies
- Strategy: actions taken by agents
- Concurrent Game Structures(CGS): KS with labeled edges
- $\langle\langle C\rangle\rangle\varphi$: coalition C has a collective strategy to enforce φ



ATL Model Checking

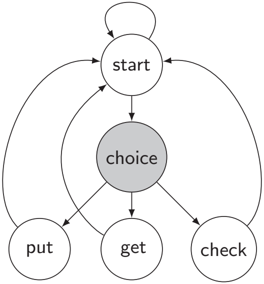
- ATL: temporal logic meets strategies
- Strategy: actions taken by agents
- Concurrent Game Structures(CGS): KS with labeled edges
- $\langle\langle C\rangle\rangle\varphi$: coalition C has a collective strategy to enforce φ



- For a CGS M and an ATL formula φ, check whether $M \models \varphi$
- An example:
  - ♠ $\langle\langle C\rangle\rangle\Diamond$pos$_1$ (C has a strategy to eventually reach pos_1)

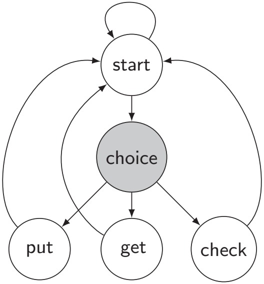
CTL Module Checking

- Models: 2-player (sys vs env) turn-based transition systems
- Environment’s behavior T': tree unfolding of M in which some envrionment subtrees are pruned.



CTL Module Checking

- Models: 2-player (sys vs env) turn-based transition systems
- Environment's behavior T': tree unfolding of M in which some envrionment subtrees are pruned.



- CTL Module Checking ($M \models_r \varphi$): Given M and a CTL formula φ, we check whether $T' \models \varphi$ for all trees T'
- Example: $M \not\models_r$ E$\Diamond$put

Module Checking vs. Model Checking

Intuition

Module checking of a CTL$^{(*)}$ formula φ can be translated to model checking of the ATL$^{(*)}$ formula $\neg\langle\langle env\rangle\rangle\neg\varphi$.

Outline

1 Introduction

2 Verification of Open Systems

3 Results

4 Conclusions

No Pruning, No Module Checking

Theorem

Standard ATL$^{(*)}$ model checking is not powerful enough to embed CTL$^{(*)}$ module checking

- Module checking uses non-dertministic strategies
- In Module Checking prunings are permanent, i.e., irrevocable.

The Result

Theorem

The simple singleton-coalition fragment of ATL/ATL with irrevocable and nondeterministic strategies is able to embed CTL/CTL* module checking.*

The Result

Theorem

The simple singleton-coalition fragment of ATL/ATL with irrevocable and nondeterministic strategies is able to embed CTL/CTL* module checking.*

Bingo ☺

Outline

1 Introduction

2 Verification of Open Systems

3 Results

4 Conclusions

Conclusions

- We formally address the relationship between CTL*/CTL module checking and ATL*/ATL model checking
- ...and show that it's not what it seemed[1].

[1] Full results presented at AAMAS'14

Conclusions

- We formally address the relationship between CTL*/CTL module checking and ATL*/ATL model checking
- ...and show that it's not what it seemed[1].

Meta-Conclusions: The Fall and Rise of Module Checking

- Module checking is worth practical investigation!
- There are several application of CTL$^{(*)}$ module checking one can investigate...

[1] Full results presented at AAMAS'14

Thank you for your attention!