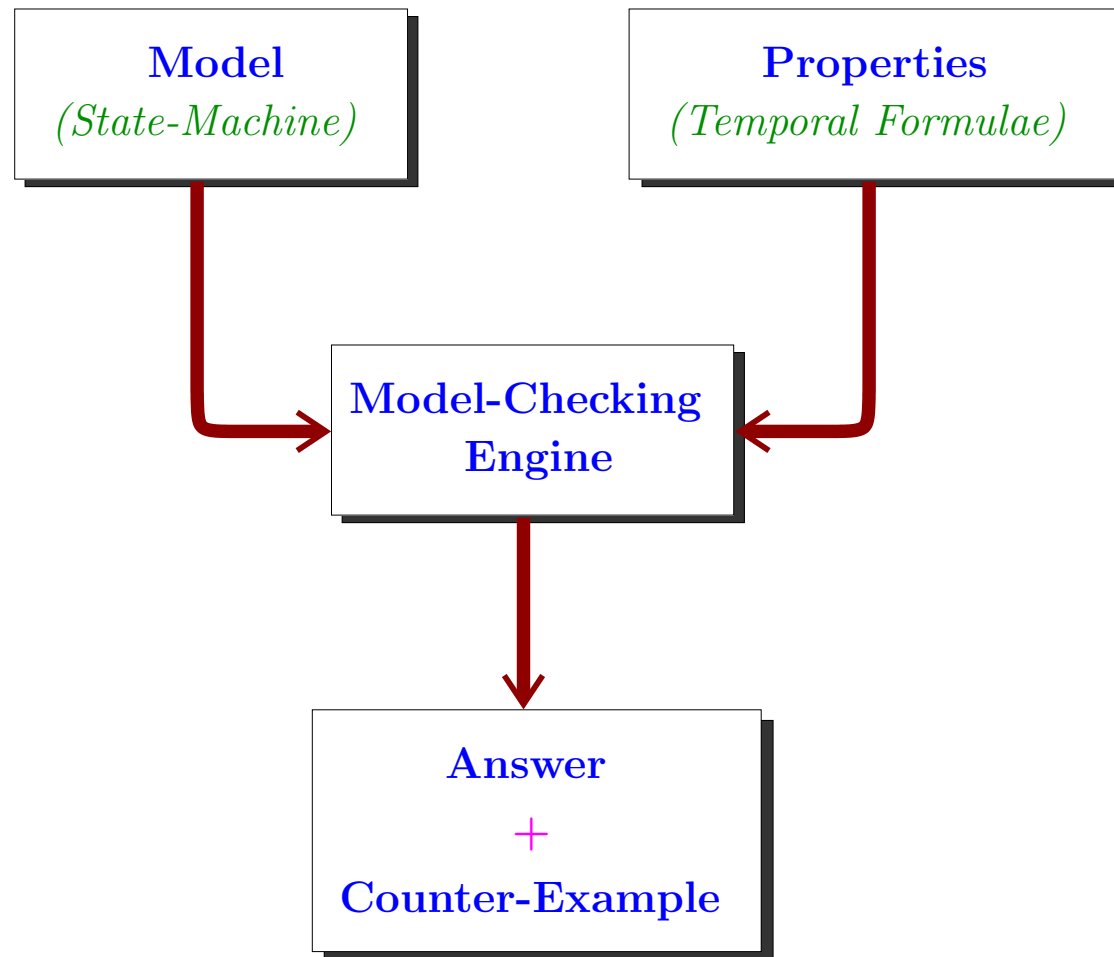


Stuttering Refinement on Partial Specifications

Shiva Nejati Arie Gurfinkel
Department of Computer Science,
University of Toronto, Canada

Email: {shiva, arie}@cs.toronto.edu

Model Checking



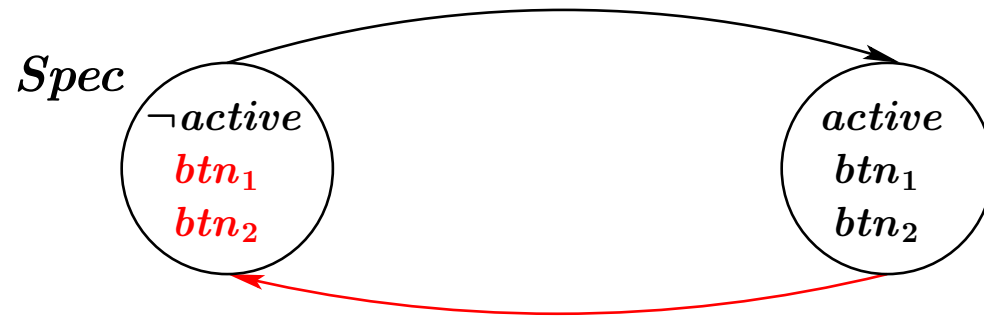
- **Disadvantage:** state explosion problem.

Goals

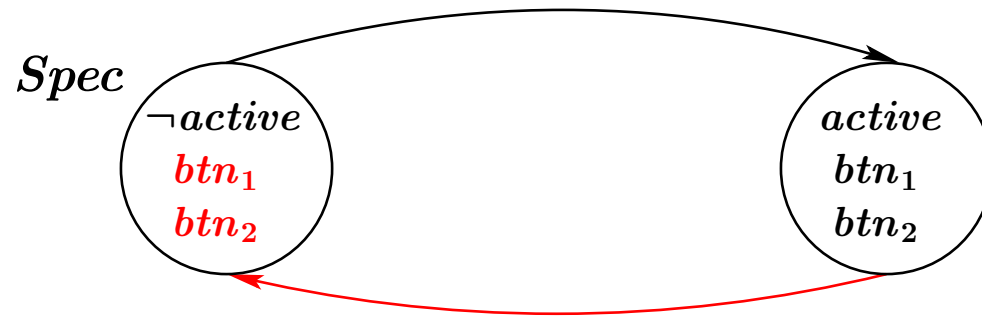
- **Apply model checking at early stages of software development.**
 - ↳ **Fairly small models.**
 - ↳ **Need for early feedback.**

- **Models at early stages of software development are partial.**
 - ↳ **Incompleteness and uncertainty.**
 - ↳ **Incremental development.**

Partial Models and Refinement Relations

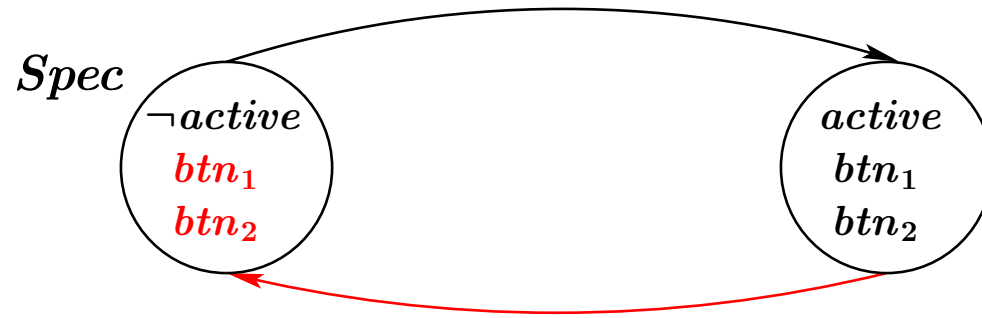


Partial Models and Refinement Relations



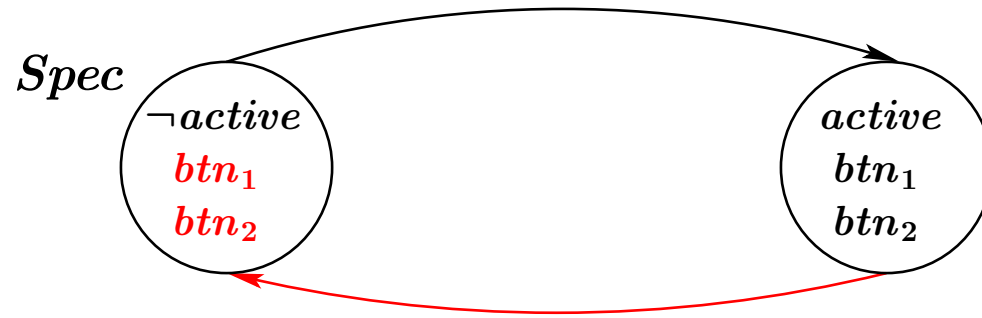
$$AG(\neg active \Rightarrow EF(active))$$

Partial Models and Refinement Relations



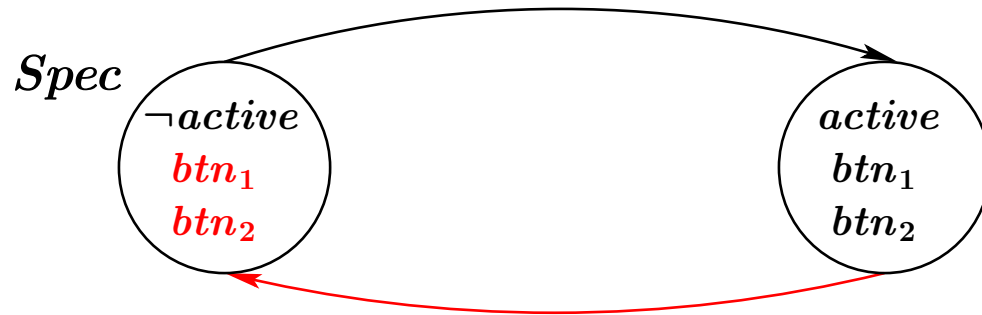
✓ $AG(\neg active \Rightarrow EF(active))$

Partial Models and Refinement Relations

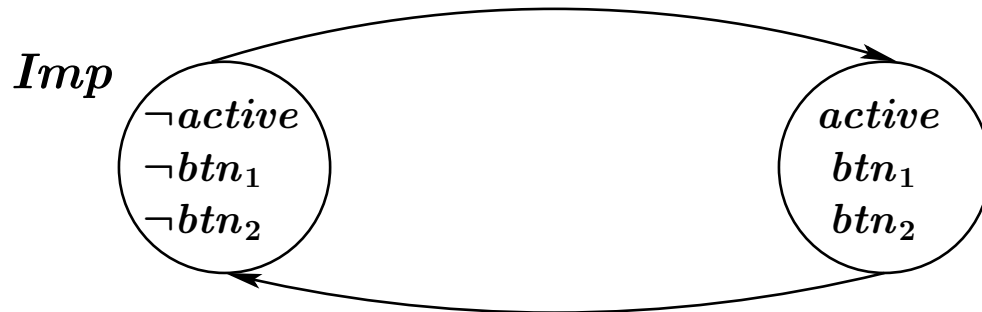


✓ $AG(\neg active \Rightarrow EF(active))$
 ✗ $AG(btn_1 \neq btn_2)$

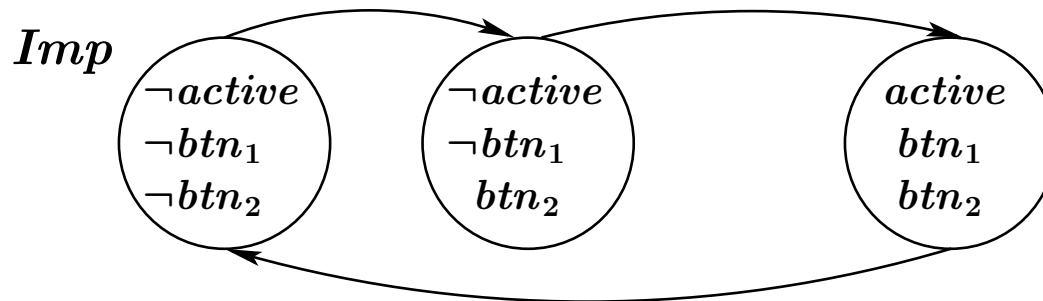
Partial Models and Refinement Relations



- ✓ $AG(\neg active \Rightarrow EF(active))$
- ✗ $AG(btn_1 \neq btn_2)$
- ? $AG(btn_1 = btn_2)$

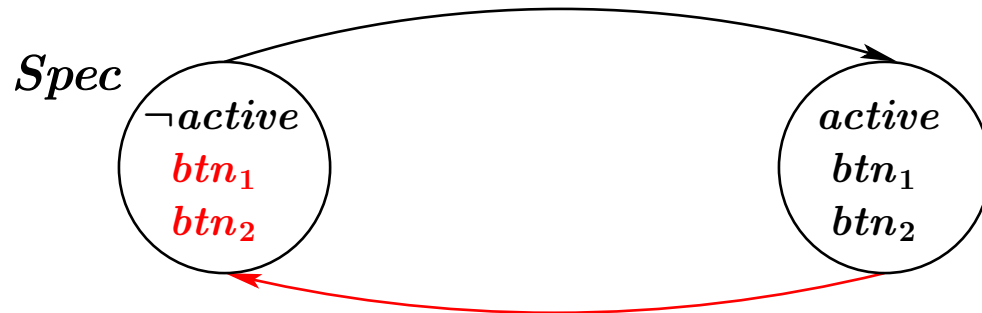


- ✓ $AG(\neg active \Rightarrow EF(active))$
- ✗ $AG(btn_1 \neq btn_2)$
- ✓ $AG(btn_1 = btn_2)$

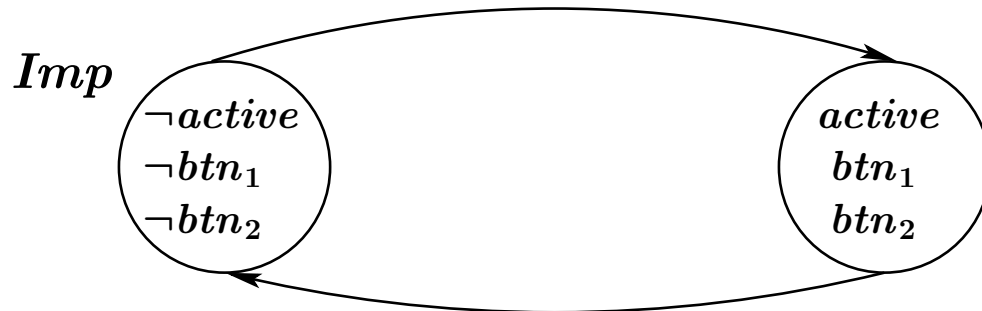


- ✓ $AG(\neg active \Rightarrow EF(active))$
- ✗ $AG(btn_1 \neq btn_2)$
- ✗ $AG(btn_1 = btn_2)$

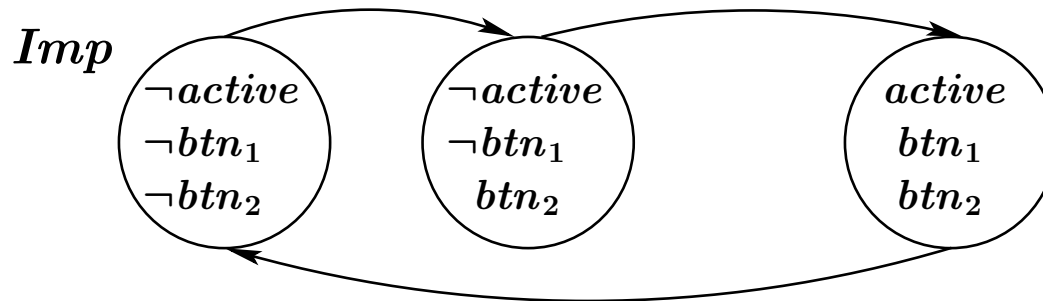
Partial Models and Refinement Relations



- ✓ $AG(\neg active \Rightarrow EF(active))$
- ✗ $AG(btn_1 \neq btn_2)$
- ? $AG(btn_1 = btn_2)$
- ✓ $EX(btn_1)$



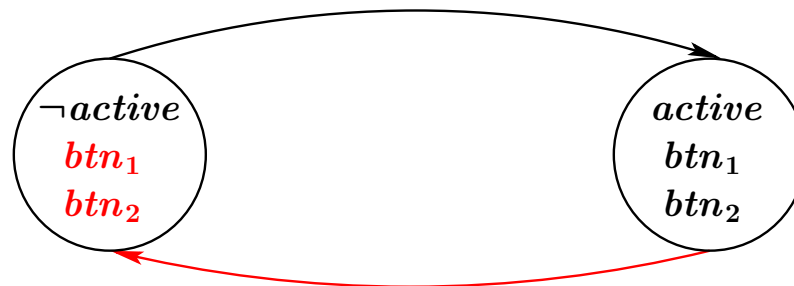
- ✓ $AG(\neg active \Rightarrow EF(active))$
- ✗ $AG(btn_1 \neq btn_2)$
- ✓ $AG(btn_1 = btn_2)$
- ✓ $EX(btn_1)$



- ✓ $AG(\neg active \Rightarrow EF(active))$
- ✗ $AG(btn_1 \neq btn_2)$
- ✗ $AG(btn_1 = btn_2)$
- ✗ $EX(btn_1)$

Stuttering Refinement [Nej03]

- Preserves CTL_X properties: universal and existential properties, no next operator.
- Achieves a good state-space reduction.
- Enables information hiding:



↳ Different variables can be hidden at different states.■

✍ We have an algorithm for computing stuttering refinement:

- Based on partition refinement algorithms [GV90].
- Polynomial in the size of the state-space.

Related Work

	state-space reduction	property preservation	information hiding
Simulation [Mil80, Par81]	✓	only universal or existential	✗
Bisimulation [Mil80, Par81]	✗	✓	✗
Refinement [LT88, HJS01]	✓	✓	limited
Stuttering equivalence [BCG88]	✓	✓	per model
Stuttering refinement	✓	✓	per state

Future Work

- **Algorithm improvement:**
 - ⇒ **Tighter complexity bounds.**
 - ⇒ **Implementation engineering.**
- **Further case studies.**
- **Use of stuttering refinement checking to speed up model checking.**
- **Fair refinement and fair stuttering refinement relation over partial models.**

References

- [BCG88] M. C. Browne, E. M. Clarke, and O. Grumberg. Characterizing finite kripke structures in propositional temporal logic. *Theoretical Computer Science*, 59(1-2):115–131, 1988.
- [GV90] J. F. Groote and F. Vaandrager. An efficient algorithm for branching bisimulation and stuttering equivalence. In *Proceedings of the seventeenth international colloquium on Automata, languages and programming*, pages 626–638. Springer-Verlag New York, Inc., 1990.
- [HJS01] M. Huth, R. Jagadeesan, and D. A. Schmidt. Modal transition systems: A foundation for three-valued program analysis. In *Proceedings of 10th European Symposium on Programming (ESOP'01)*, volume 2028 of *Lecture Notes in Computer Science*, pages 155–169. Springer, 2001.
- [LT88] K.G. Larsen and B. Thomsen. A modal process logic. In *Proceedings of 3rd Annual Symposium on Logic in Computer Science (LICS'88)*, pages 203–210. IEEE Computer Society Press, 1988.
- [Mil80] R. Milner. A calculus of communicating systems. LNCS 92. Springer-Verlag, 1980.
- [Nej03] S. Nejati. Refinement of partial specifications, 2003.
- [Par81] D. Park. Concurrency and automata on infinite sequences. LNCS 104, pages 167–183. Springer-Verlag, 1981.