Verification of Parameterized System Does Processes' Global Knowledge Affect Decidability?

Zhang Ting
tingz@cs.stanford.edu
Stanford University

November 22, 2000

Overview

- Part I: Introduction
  1. Negative Results
  2. Positive Results
- Part II: Knowledge Framework
  1. Introduction to Knowledge Framework
  2. Observational Knowledge
- Part III: Undecidabilities
  1. Bidirectional Network
  2. Unidirectional Network with Different Orientations
  3. Unidirectional Network with Uniformly Orienteds
- Conclusion

Introduction

- **Negative Results**
  - Generally language of (P, ϕ) is Π^0_1-complete [KRA86].
  - Unidirectional ring of finite machines [Suz88, EN95].
  - Global properties (in IPLTL) in systems without communication network [GS92].
- **Positive Results**
  - Reachability Analysis
    * Explicit State Space [GS92]
      Features: No communication network, anonymous communications/synchronizations.
      Complexity: 1. EXSPACE-hard for local properties with a control process. 2. Polynomial for local properties without control processes. 3. Undecidable for general global properties. 4. EXSPACE-hard for some global properties (e.g., mutual exclusion, deadlock detection).
      Cons: A very limited computation model.
    * Abstract State Space [EN96]
      Features: Synchronous global execution.
      Complexity: PSPACE-complete
      Cons: A restricted computation model. E.g., mutual exclusion is not implementable.
    * Symbolic State Space [KMM$^+$97, PAAN99]
      Features: Symbolic forward reachability analysis with

regular language representation of system configurations.
Cons: Semi-procedure, linearly ordered processes, safety properties.

Properties of reachability analysis: Not many practical computation models fall in this category and complexity is too high.

- Symmetry Reduction
  * Token rings [EN95]
    Features: Reduce correctness checking to a ring of small size. Properties are $\wedge_i\ g_i$, $\wedge_i\ g_{i,i+1}$, $\wedge_{i\neq j}\ g_{ij}$ etc.
    Complexity: Polynomial.
    Cons: Only allows unary token value.
  * Scalar data type [ID96]
  * Replicated component ID [ID99]
    Features: Abstract away ordering information from the original state space.
    Cons: No correctness proofs for temporal properties.
  * Quotient structures. [CFJ93, ES93, ES97]
    Features: Explore symmetry both in systems and temporal properties with fairness considered.

  Properties of symmetry methods: State space reduction doesn't provide correctness proofs for systems of arbitrary size.
- Induction
  * Network/Process invariant [WL89, KM89]
    Features: Completely algorithmic with the

network(process) invariant presence.
Cons: A (finite) network/process invariant doesn't always exist (incompleteness).

∗ Global state projection [MCBG89, CG87]
Features: Establish a bisimulation relation on process closures between networks of different sizes.
Cons: Restrictions on logic. E.g., no $next$ operator.

∗ Network grammars and regular languages [SG89, CGJ95]
Features: 1. Represent infinite families of processes in network grammar. 2. Express properties in regular languages.

∗ Dynamic induction on diagrams [MS99]
Features: Induction on process indices whose partial orders are state-dependent.

Properties of induction methods: Constructing suitable inductive structures needs human ingenuity.

- Negative Result Characteristics: Shared global configuration knowledge or existence of control processes.
- Question: No global configuration knowledge $+$ no control processes $\leadsto$ decidability?
- Answer: Negative.

Knowledge Framework

Knowledge formalism in propositional modal logic [RFV92], [RFV95].

Let $\mathcal{M}_n = \langle W, \pi, \mathcal{K}_1, \ldots, \mathcal{K}_n \rangle$ be a standard Kripke structures with *possible world* relations $\mathcal{K}_i$. π is a mapping function which associates each $w \in W$ a set of primitive propositions. Let $\mathcal{L}_n$ be a formal language with n *knowledge operators* K_i. Semantics of $\mathcal{M}_n$ is:

- $\langle M, w \rangle \models \phi$ iff ϕ is a primitive proposition and $\phi \in \pi(w)$.
- $\langle M, w \rangle \models \neg\phi$ iff $\langle M, w \rangle \not\models \phi$.
- $\langle M, w \rangle \models \phi \wedge \psi$ iff $\langle M, w \rangle \models \phi$ and $\langle M, w \rangle \models \psi$.
- $\langle M, w \rangle \models K_i\phi$ iff $\langle M, w' \rangle \models \phi$ for all w' where $(w, w') \in \mathcal{K}_i$.

Let $\mathcal{K}_i$ to be equivalent relations and the $\mathcal{M}_n$ becomes a $\mathbf{S5}$ structure. With the following axioms (marked in $\mathbf{S5}$ notation) $\mathcal{L}_n$ is sound and complete axiomization with respect to $\mathcal{M}_n$ [RFV95].

- (A1) All tautologies substitution of propositional calculus (Tautology)

- (A2) [**K**] $(K_i\phi \wedge K_i(\phi \Rightarrow \psi)) \Rightarrow K_i\psi$ (Distribution Axiom)
- (A3) [**T**] $K_i\phi \Rightarrow \phi$ (Knowledge Axiom)
- (A4) [**4**] $K_i\phi \Rightarrow K_iK_i\phi$ (Positive Introspection Axiom)
- (A5) [**5**] $\neg K_i\phi \Rightarrow K_i\neg K_i\phi$ (Negative Introspection Axiom)
- (R1) From ϕ and $\phi \Rightarrow \psi$ infer ψ (Modus Ponen)
- (R2) From ϕ infer $K_i\phi$ (Knowledge Generalization)

Problem: (R2) causes *logical omniscience* problem. I.e., a process knows all logical truth.

Solution: Observational Knowledge.
"Knows ϕ" = "ϕ is true" + "ϕ is observed"

Assumptions: Processes' knowledge is *cumulative.* I.e., processes have *"perfect recall"* ability.

Reasons: Sharpen the negative result.

Imagine that each process is associated with an *external agent* which is observing the computations of a process. If such an *external observant* can not find a global configuration property only through its observation of a process's computation, it is reasonable to believe that a finite state process can not find it either.

Observational Knowledge

Let $\mathcal{L}_n^o$ augmented with *observational knowledge operator* K_i^o for $i \in \{1, \ldots, n\}$. Let $\mathcal{M}_n^o = \langle W, \pi_1, \ldots, \pi_n, \pi, \mathcal{K}_1, \ldots, \mathcal{K}_n, \rangle$ be the semantics model. $\mathcal{M}_n^o$ has new mapping functions π_i which associate each world (state) with a set of primitive propositions *"observed"* in the state. Auxiliary semantics $\models_i$ for each $i \in \{1, \ldots, n\}$ are:

- (S0) $\langle M, w\rangle \models_i p$ iff $p \in \pi_i(w)$ where p is a primitive proposition.
- (S1) $\langle M, w\rangle \models_i \neg\phi$ implies $\langle M, w\rangle \not\models_i \phi$.
- (S2) $\langle M, w\rangle \models_i \phi \wedge \psi$ iff $\langle M, w\rangle \models_i \phi$ and $\langle M, w\rangle \models_i \psi$.
- (S3) $\langle M, w\rangle \models_i \phi \vee \psi$ iff $\langle M, w\rangle \models_i \phi$ or $\langle M, w\rangle \models_i \psi$.
- (S4) $\langle M, w\rangle \models_i \neg K_i\phi$ iff $\langle M, w\rangle \not\models_i K_i\phi$.
- (S5) $\langle M, w\rangle \models_i \neg K_i^o\phi$ iff $\langle M, w\rangle \not\models_i K_i^o\phi$.
- (S6) $\langle M, w\rangle \models_i K_i\phi$ iff $\langle M, w'\rangle \models_i \phi$ for all w' where $(w, w') \in \mathcal{K}_i$.
- (S7) $\langle M, w\rangle \models_i K_i^o\phi$ iff $\langle M, w'\rangle \models_i \phi$ for all w' where $(w, w') \in \mathcal{K}_i$.

Observational Knowledge (Con'd)

The semantics for formula $K_i^o \phi$ is: for all $(w, w') \in \mathcal{K}_i$,

$$\langle M, w\rangle \models K_i^o \phi \text{ iff } \langle M, w'\rangle \models_i \phi \text{ and } \langle M, w'\rangle \models \phi$$

With additional *observation operator* O_i introduced, semantics definition can be simplified as follows:

$$\langle M, w\rangle \models O_i \phi \text{ iff } \langle M, w'\rangle \models_i \phi \text{ for all } w' \text{ where } (w, w') \in \mathcal{K}_i$$

$$\langle M, w\rangle \models K_i^o \phi \text{ iff } \langle M, w\rangle \models O_i \phi \text{ and } \langle M, w\rangle \models K_i \phi$$

Parameterized System with Unwareness

Let $\mathcal{C}_k$ be the class of parameterized systems which can *observationally know* global knowledge. Let $\bar{\mathcal{C}_k}$ be the complement. We identify a set of parameterized systems $\mathcal{P}$ and show $\mathcal{P}$ in $\bar{\mathcal{C}_k}$. $\langle \mathbf{P}, \mathbf{N} \rangle$ where $\mathbf{P}$ is a process template and $\mathbf{N}$ is an index set. $\mathcal{P}$ consists of $\langle \mathbf{P}, \mathbf{N} \rangle$ with the following properties:

1. All processes are anonymous. I.e., process ID is just an external reference for our discussion. It isn't stored in the local state of a process.
2. There is no system-wide non-constant variable.
3. Each process is a completely identical finite state transition system.
4. Each process communicates with a constant number of processes, called its *neighbors*. Channels are identified and communication are synchronized or asynchronous with order preserving and no loass. E.g., in implementation, it is enough to let each process have local variables in binary domains which is only readable by its *neighbors*.

Let P^m denote a system instance of size m. Think that all processes are organized on a ring of size m. All arithmetics on process indices are defined in $[m]$, (roughly). Finally, note that it is impossible to do leader election in such a system [Ang80].

Graph Embedding

Definition 1. *Given two graphs F and G, F is $embedded$ in G (respectively, G is $embedding$ F) if the following two condition hold:*

- *There is a homomorphism f from G to F ($G \xrightarrow{f} F$) such that $(u, v) \in E(G)$ implies $(f(u), f(v)) \in E(F)$ for $u, v \in V(G)$.*
- *For any node v in G, f is an isomorphism between $N_g(v)$ and $N_F(f(v))$ ($\forall\ v \in V_g, N_g(v) \xleftrightarrow{f} N_F(f(v))$)*

Example:

$$f = \{\langle 1, A\rangle, \langle 2, B\rangle, \langle 3, C\rangle, \langle 4, A\rangle, \langle 5, B\rangle, \langle 6, C\rangle\}$$

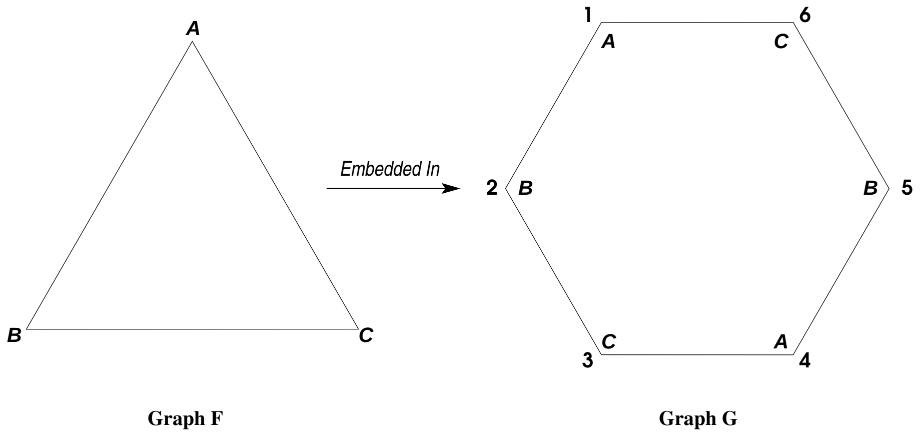

Figure 1: A Graph Embedding Example

Graph Embedding (Con'd)

Proposition 1. *Let P^m denote an instance of a system $\langle \mathbf{P}, \mathbf{N} \rangle$ in $\mathcal{P}$. Let G^m denote the communication network graph of P^m. Given two graphs G^{m1} and G^{m2} which both embeds G^m, then G^{m1+m2} embeds G^m.*

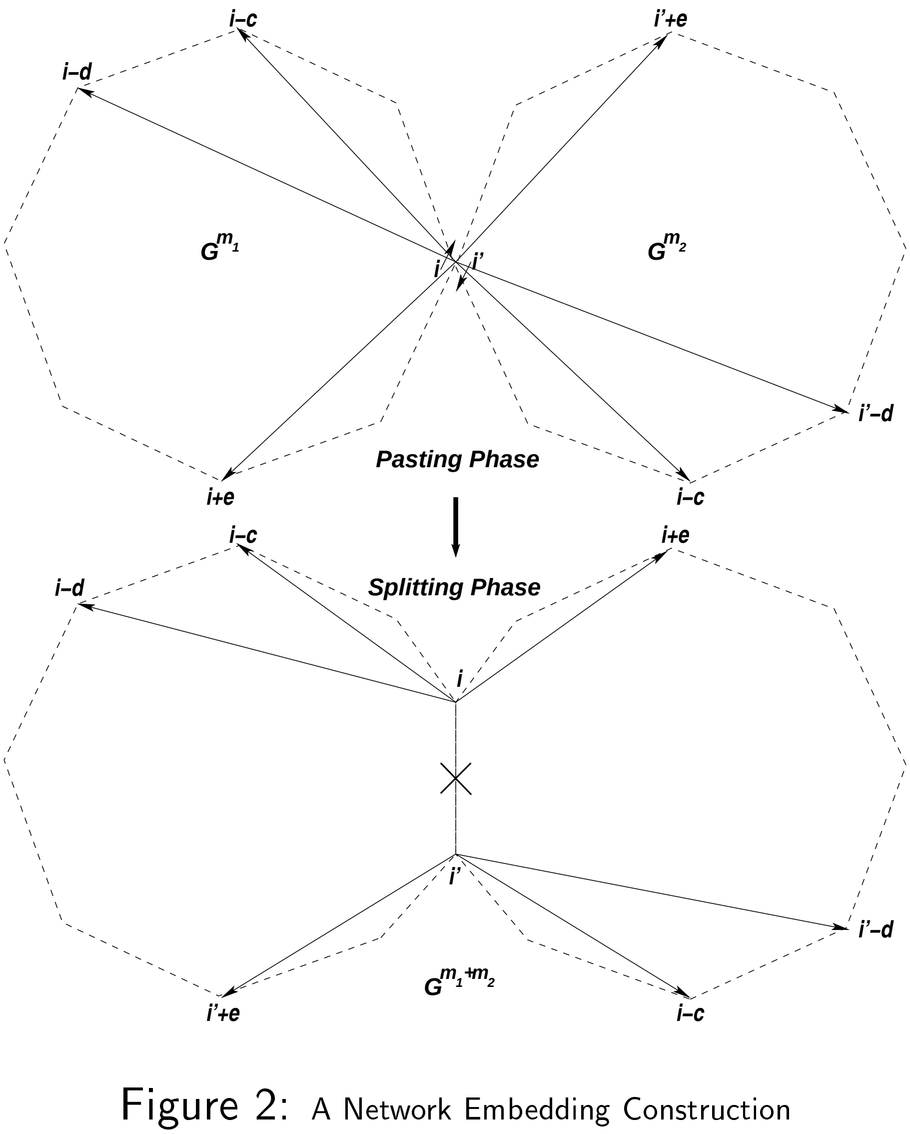

Figure 2: A Network Embedding Construction

Proposition 2. *Let G^m defined as before. G^{km} embeds G^m for all $k \in \mathbf{N}$.*

Definition 2. *Let σ^{m_1} and σ^{m_2} be computations in P^{m_1} and P^{m_2} respectively. We say σ^{m_1} is embedded in σ^{m_2} if there exists a onto index mapping f from $\{1, \ldots, m_2\}$ to $\{1, \ldots, m_1\}$ such that the computation projection $\sigma^{m_1}_{f(i)}$ is a stuttering equivalent to computation projection $\sigma^{m_2}_i$. P^{m_1} is computationally embedded in P^{m_2} if any computation σ^{m_1} in P^{m_1} is embedded in some computation σ^{m_2} in P^{m_2}.*

Proposition 3. *Given two system P^{m_1} and P^{m_2} with communication graph G^{m_1} and G^{m_2} respectively. If G^{m_1} is embedded in G^{m_2}, the computation of P^{m_1} is computationally embedded in P^{m_2}.*

Proposition 4. *Let ϕ_g and ψ_g denote two inconsistent properties. If P^{m_1} is computationally embedded in P^{m_2}, then P^{m_1} consistently observes ϕ_g implies P^{m_2} doesn't consistently observe ψ_g.*

Theorem 1. *$\mathcal{P}$ is in $\bar{\mathcal{C}}_k$.*

[Proof Sketch]
Let P^m denote a system instance of size m. Apply graph embedding techniques (as in [Ang80]) with the observation that P^m is embedded in P^{km} where $k > 0$.

Undecidabilities

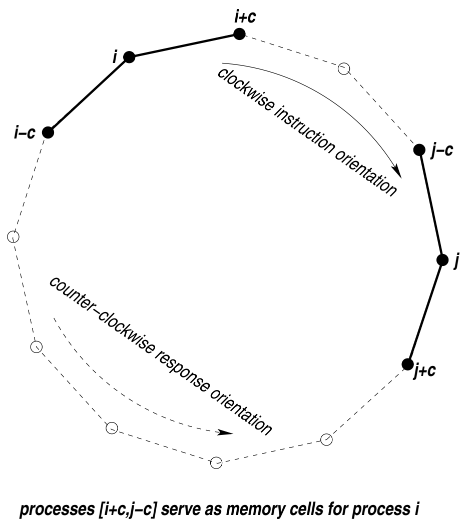

Figure 3: A Two Register Machine Simulation in A Bidirectional Network

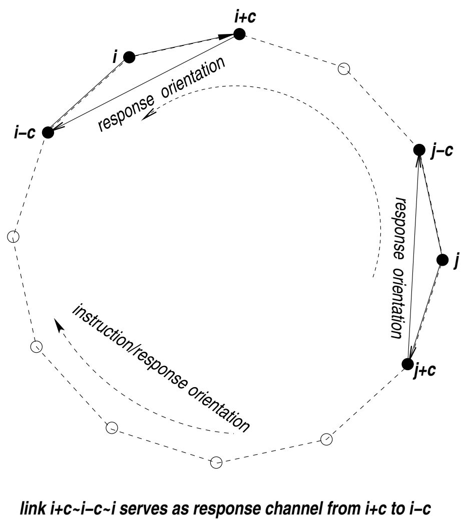

Figure 4: A Two Register Machine Simulation in A Unidirectional Network

Tag Systems

A *tag system* T is a tuple $\langle \Sigma, \Gamma, w_0, P \rangle$ where Σ is a finite alphabet, Γ a finite set of production rules, w_0 is an initial word on Σ, a nd P is a natural number, called *deletion number*. Let α_i be a letter of Σ for $i \in \{1, \ldots, |\Sigma|\}$. Each production rule τ_i is of form $\alpha_i \longrightarrow \beta_i$ where β_i is a word on Σ. If a word W on Σ beginning with letter α_i, then application of rule τ_i is to delete the beginning P letters and appending word β_i to the end.

Characteristic: production application only depends on the head letter, and hence only one rule can be applied at any time.

Example: $T = \langle \Sigma, \Gamma, w_0, P \rangle$ where

$$\Sigma = \{0, 1\} \textit{ and } \Gamma = \{0 \to 00, 1 \to 1101\}$$

$$w_0 = 100 \textit{ and } P = 3$$

$100 \to 1101 \to 11101 \to 011101 \to 10100 \to 001101 \ldots$

Definition 3. *T is halting if production sequence of w_0 is finite. T is periodic if production sequence of w_0 is infinitely repeated at some point afterwards.*

Undecidabilities: Cont'd

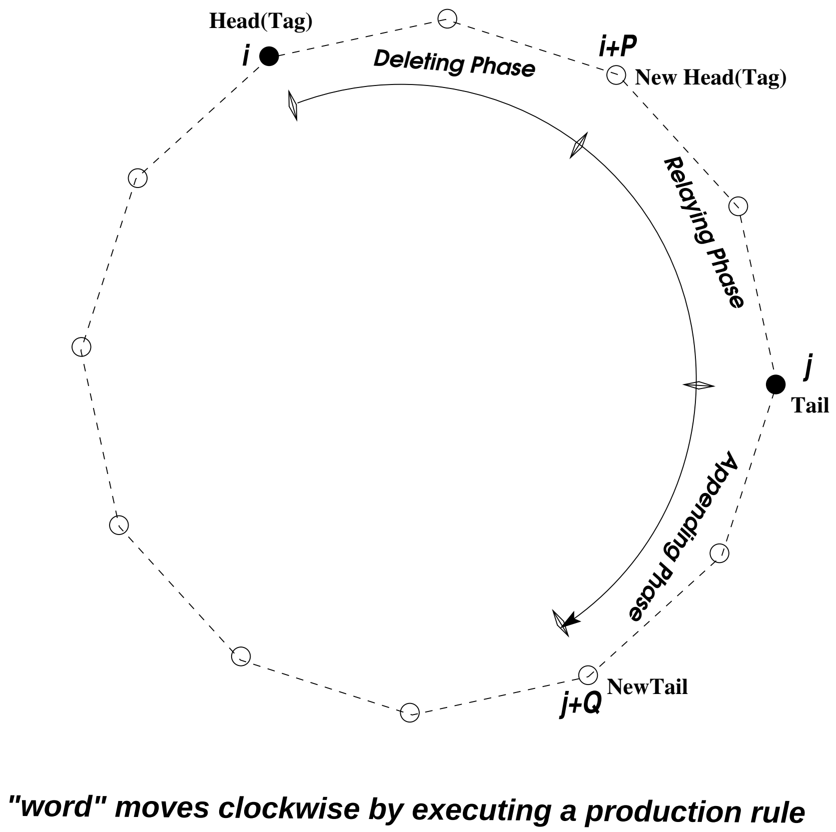

Figure 5: A Tag System Simulation in A Uniformly Oriented Network

Theorem 2. *[Min61] Tag systems are $universal$ with $P = 2$.*

Proposition 5. *A tag system T is not $periodic$ iff eventually forever all processes are out of control state.*

Proposition 6. *A tag system T is not $halting$ iff no process enter $halting$ state forever.*

Conclusion

- Parameterized system verification is undecidable as long as there is an identified communication path with unbounded length.
- Process's global knowledge doesn't affect decidability.

References

[Ang80] D. Angluin. Local and global properties in networks of processes. In *Symposium on Theory of Computing*, pages 82–93, 1980.

[CFJ93] E.M. Clarke, T. Filkorn, and S. Jha. Exploiting symmetry in temporal logic model checking. In *Computer Aided Verification*, vol. 697 of *Lecture Notes in Computer Science*, pages 450–462. Springer-Verlag, June 1993.

[CG87] E.M. Clarke and O. Grumberg. Avoiding the state explosion problem in temporal logic model checking algorithms. In *Symposium on Principles of Distributed Computing*, pages 284–303, 1987.

[CGJ95] E.M. Clarke, O. Grumberg, and S. Jha. Verifying parameterized networks using abstraction and regular languages. In I. Lee and S. Smolka, editors, *Concurrency Theory*, vol. 962 of *Lecture Notes in Computer Science*, pages 395–407. Springer-Verlag, 1995.

[EN95] E.A. Emerson and K.S. Namjoshi. Reasoning about rings. In *Symposium on Principles of Programming Languages*, pages 85–94, 1995.

[EN96] E.A. Emerson and K.S. Namjoshi. Automatic verification of parameterized synchronous systems. In R. Alur and T.A. Henzinger, editors, *Computer Aided Verification*, vol. 1102 of *Lecture Notes in Computer Science*, pages 87–98, 1996.

[ES93] E.A. Emerson and A.P. Sistla. Symmetry and model checking. In *Computer Aided Verification*, vol. 697 of *Lecture Notes in Computer Science*, pages 463–478, June 1993.

[ES97] E.A. Emerson and A.P. Sistla. Utilizing symmetry when model checking under fairness assumptions: An automata-theoretic approach. *ACM Transactions on Programming Languages and Systems*, 19(4):617–638, July 1997.

[GS92] S.M. German and A.P. Sistla. Reasoning about systems with many processes. *Journal of the ACM*, 39:675–735, 1992.

[ID96] C.N. Ip and D.L. Dill. Better verification through symmetry. *Formal Methods in System Design*, 9(1/2):41–75, August 1996.

[ID99] C.N. Ip and D.L. Dill. Verifying systems with replicated in murphi. *Formal Methods in System Design*, 14(3):273–310, May 1999.

[KM89] R.P. Kurshan and K.L. McMillan. A structural induction theorem for processes. In *Symposium on Principles of Distributed Computing*, pages 239–247. ACM Press, 1989.

[KMM+97] Y. Kesten, O. Maler, M. Marcus, A. Pnueli, and E. Shahar. Symbolic model checking with rich assertional languages. In *Computer Aided Verification*, vol. 1254 of *Lecture Notes in Computer Science*, pages 424–435. Springer-Verlag, June 1997.

[KRA86] D.K. Krzysztof R. Apt. Limits for automatic verification of finite-state concurrent systems. *Information Processing Letters*, 22(6):307–309, 30 May 1986.

[MCBG89] E.M.C. Michael C. Browne and O. Grumberg. Reasoning about networks with many finite state processes. *Information and Computation*, 81(1):13–31, April 1989.

[Min61] M.L. Minsky. Recursive unsolvability of post's problem of tag and other topics in theory of turing machines. *The Annals of Mathematics*, 74(3):437–455, November 1961.

[MS99] Z. Manna and H.B. Sipma. Verification of parameterized systems by dynamic induction on diagrams. In N. Halbwachs and D. Peled, editors, *Computer Aided Verification*, vol. 1633 of *Lecture Notes in Computer Science*, pages 25–43, Trento, Italy, July 1999. Springer-Verlag.

[PAAN99] B.J. Parosh Aziz Abdulla, Ahmed Bouajjani and M. Nillson. Handling global conditions in parameterized system verification. In N. Halbwachs and D. Peled, editors, *Computer Aided Verification*, vol. 1633 of *Lecture Notes in Computer Science*, pages 134–145, Trento, Italy, July 1999. Springer-Verlag.

[RFV92] J.Y.H. Ronald Fagin and M.Y. Vardi. What can machines know? on the properties of knowledge in distributed systems. *Journal of the ACM*, 39:328–376, 1992.

[RFV95] Y.M. Ronald Fagin, Joseph Y. Halpern and M.Y. Vardi. *Reasoning about Knowledge*. The MIT Press, Cambridge, Massachusetts, USA, 1995.

[SG89] Z. Shtadler and O. Grumberg. *Network Grammars, Communication Behaviors and Automatic Verification*, vol. 407, pages 151–165. Springer-Verlag, 1989.

[Suz88] I. Suzuki. Proving properties of a ring of finite-state machines. *Information Processing Letters*, 28(4):213–214, 29 July 1988.

[WL89] P. Wolper and V. Lovingfosse. Verifying properties of large sets of processes with network invariants. In J. Sifakis, editor, *Computer Aided Verification*, vol. 407 of *Lecture Notes in Computer Science*, pages 68–80. Springer-Verlag, 1989.