

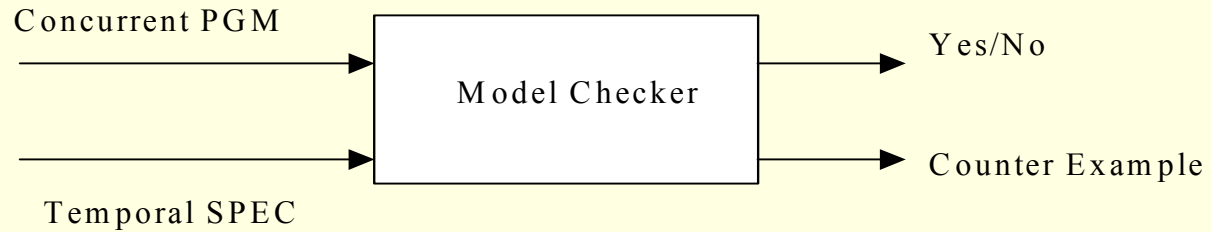
Symmetry Reductions.

A. Prasad Sistla

University Of Illinois at Chicago



Model-Checking



- Approach
 - Build the global state graph
 - Algorithm to check correctness
- Applications
 - Concurrent Programs.
 - Protocols.
 - Circuits.

Bottleneck: State Explosion

Has only been used for small size problems.

- number of states grows exponentially.
- Techniques to contain state explosion
 - Symbolic Model Checking (BDDs)
 - Stubborn Sets/Sleep Sets
 - Symmetry
(Due to identical/similar processes)

Outline

- Symmetry & Quotient Structure
 - Program Symmetry
 - Formula Symmetry
 - Quotient Structure
 - State Symmetry
- Annotated Quotient Structure.
 - Fairness
- SMC – An implemented system
- Reduced Symmetry & Assymetry.
 - Guarded quotient Strucure.
 - Formula Decomposition
 - Subformula tracking.

Model : Shared Variable

- Notation

- Variable name- $X_{i,j}$
(denotes a variable shared by processes i, j)

- Program : A set of processes.

- Process : A set of guarded commands.

- Process \mathcal{K}_i ::

- [$LC_i = 0 \wedge F_{i,i+1} = 1 \rightarrow F_{i,i+1} := 0, LC_i := 1;$
- $LC_i = 1 \wedge F_{i,i-1} = 1 \rightarrow F_{i,i+1} := 0, LC_{i,i+1} := 2;$
- $LC_i \rightarrow F_{i,i+1} := 1, F_{i,i+1} := 1, LC_i := 0;]$

- Program \mathcal{K} :: $\mathcal{K}_1 || \mathcal{K}_2 || \dots || \mathcal{K}_n$

Program Symmetry

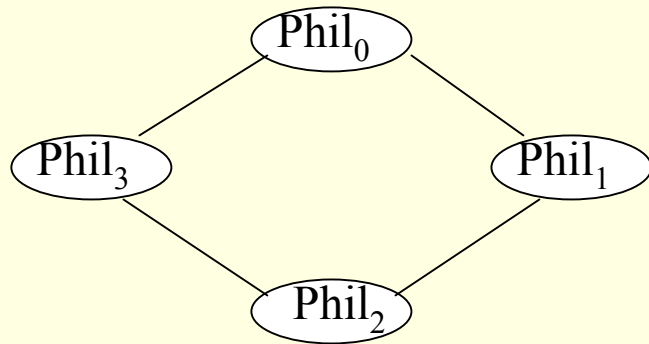
- $\mathcal{K} :: \mathcal{K}_1 \parallel \mathcal{K}_2 \parallel \dots \parallel \mathcal{K}_n$
- \parallel is commutative and associative
- $\mathcal{I} = \text{Index set} = \{1, 2, \dots, n\}$
- For a permutation π on \mathcal{I} ,
- define $\pi(\mathcal{K}_i)$ - Process obtained by changing indices of variables according to π
($X_{i,j}$ changed to $X_{\pi(i), \pi(j)}$)
- $\text{Aut } \mathcal{K} = \{ \pi \mid \pi(\mathcal{K}) = \mathcal{K} \}$

- Global State s : Assignment of values to variables.
- Global State Graph : $\mathcal{M} = (\mathcal{S}, \mathcal{R})$
 - \mathcal{S} - Set of global states
 - $(s, t) \in \mathcal{R}$ iff t is obtained from s by executing a single step of some process.
(interleaved semantics)
- Interested in Symmetry of \mathcal{M} :

- s: global state.
- π : permutation on \mathcal{I} ,
 - $\mathcal{I} = \{0, 1, \dots, n-1\} = \text{Process Indices.}$
- $\pi(s)$ is a global state in which
 - variable $X_{\pi(i), \pi(j)}$ gets the value of $X_{i,j}$ in s
- $\text{Aut}(\mathcal{M}) = \{\pi \mid \pi(\mathcal{M}) = \mathcal{M}\}$
 - $\text{Aut } \mathcal{M}$ is a group
- Lemma: $\text{Aut } \mathcal{K} \subseteq \text{Aut } \mathcal{M}$

Examples:

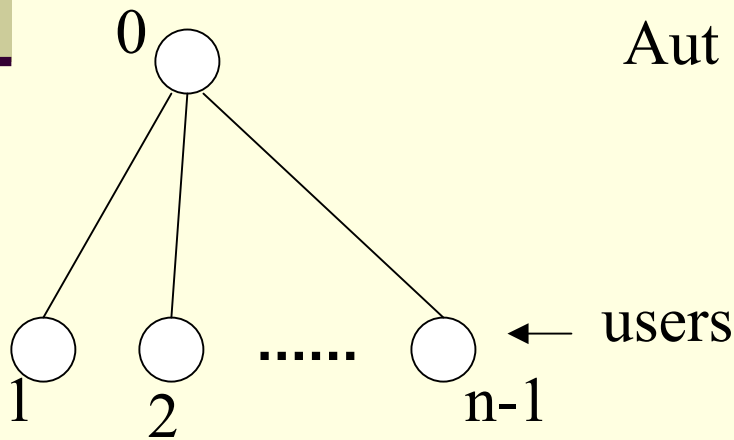
- For Symmetric soln of Dining Phil. Problem :



$\text{Aut } \mathcal{M} = \{ \pi : \pi \text{ is a circular perm.} \}$

$\text{Aut } (\mathcal{K}) = \text{Aut } (\mathcal{M})$

- For the Resource controller Prog:



$\text{Aut } \mathcal{M} = \{ \pi : \pi(0) = 0 \}$



π can permute the users.

0 is the controller process

Logics of Programs

CTL*:

Temporal operators: F, G, X, U

$F P$: *eventually* P

$G P$: *always* P

$X P$: *nexttime* P

$P U Q$: P *until* Q

Path Quantifiers:

A – for all paths

E – for some path

$AG(P)$: Invariance

$AF(P)$: Inevitability of P

P : Basic assertion, uses indexed variables.

Ex: $LC_i = L, X_{i,j} > 0$

f : a formula in CTL*

$\pi(f)$ obtained by changing indices of variables according to π .

Symmetry of formulas:

$$\text{Aut } f = \{ \pi : \pi(f) \equiv f \}$$

We use a subgroup of $\text{Aut } f$ called $\text{Auto } f$.

$$\text{Auto } f = \bigcap (\text{Aut } p)$$

p is a maximal
prop. subformula of f .

Examples:

$$f = AG \left((T_1 \vee T_2) \rightarrow AF(C_1 \vee C_2) \right)$$

Global liveness for a mutual exclusion problem with two processes.

T_i : Process i is in trying mode.

C_i : Process i is in critical section.

$\text{Auto}(f) = \text{Sym } \mathcal{I} = \text{Set of all permtns. } \mathcal{I} = \{1, 2\}$

$$g = \bigwedge_{i=1,2} AG (T_i \rightarrow AF C_i)$$

$$\text{Aut}(g) = \text{Sym } \mathcal{I}, \quad \text{Auto}(g) = \{\text{Id}\}$$

Quotient Structure:

$\mathcal{M} = (\mathcal{S}, \mathcal{R})$ – structure

f – CTL* formula

$G \subseteq \text{Aut } \mathcal{M} \cap \text{Auto } f$

Equivalence relation \equiv_G on S

$s \equiv_G t$ iff $\exists \pi \in G$ such that $\pi(s) = t$

$\mathcal{M}/G = (\mathcal{S}^+, \mathcal{R}^+)$: Quotient Structure.

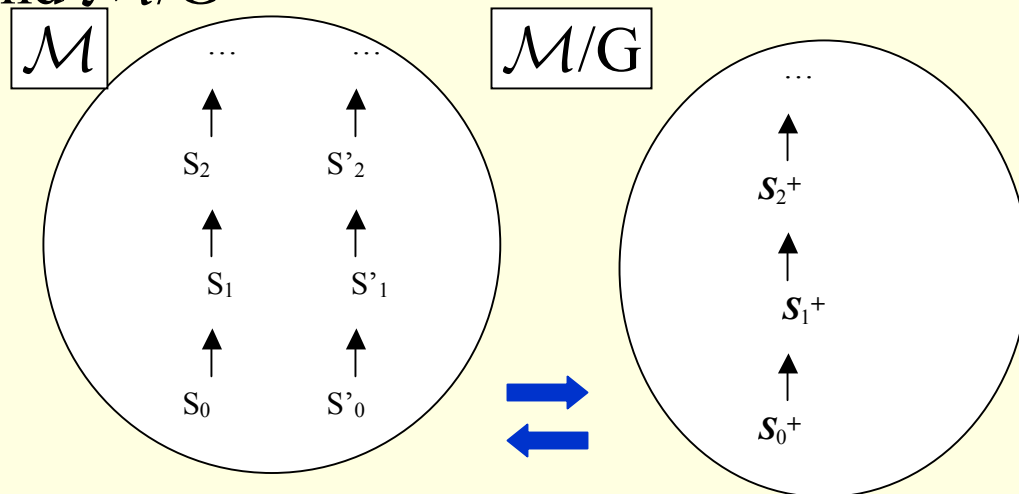
\mathcal{S}^+ has one representative for each equiv. class.

$(s^+, t^+) \in \mathcal{R}^+$ iff for some $s \equiv_G s^+$, $t \equiv_G t^+$

$(s, t) \in \mathcal{R}$

Correspondence Lemma

- There is a bidirectional correspondence between paths of \mathcal{M} and \mathcal{M}/G



- $(s_0, s_1, \dots, s_i, \dots) \in \mathcal{M} \Rightarrow (s_0^+, s_1^+, \dots, s_i^+, \dots) \in \mathcal{M}/G$
- $(s_0^+, s_1^+, \dots, s_i^+, \dots) \in \mathcal{M}/G \Rightarrow \forall s_0^| \equiv_G s_0^+, \exists \text{ path in } \mathcal{M} (s_0^|, s_1^|, \dots, s_i^|, \dots) \text{ such that } s_i^| \equiv_G s_i^+$

Main Theorem

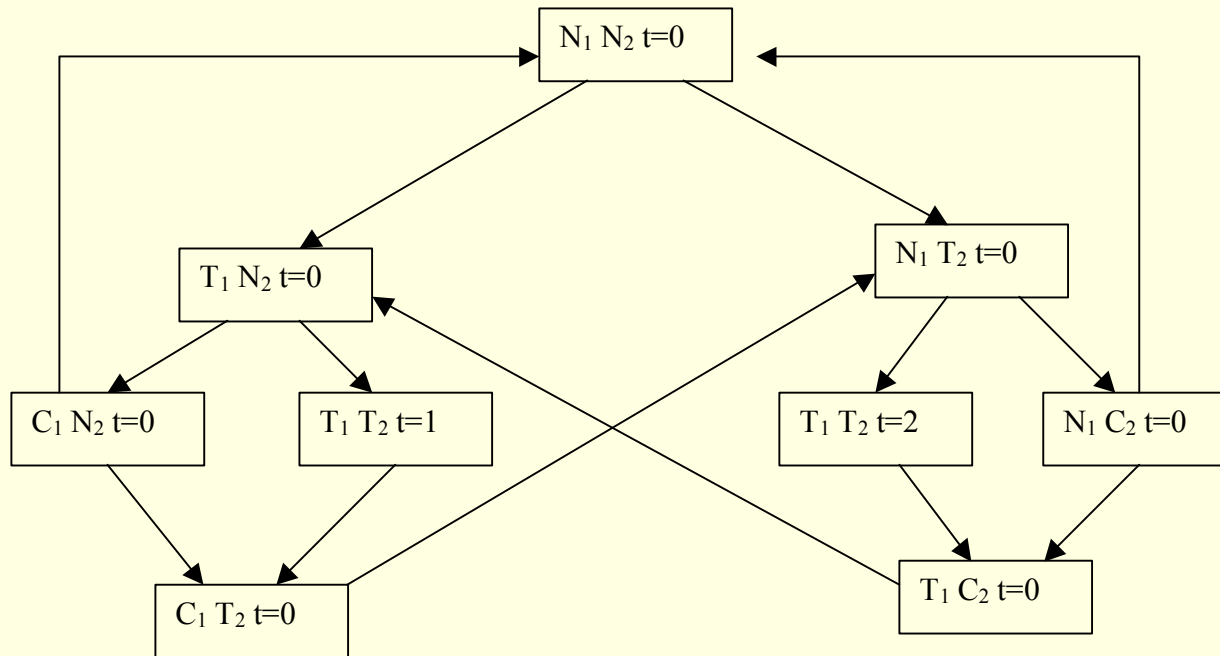
- For any $s \in \mathcal{S}$,
- $\mathcal{M}, s \models f$ iff $\mathcal{M}/G, s^+ \models f$
- (Ip & Dill 93, CFG 93, ES 93)
- Proof: Uses induction on f and the *corr.* Lemma.

● Examples

Dining Phil. Problem

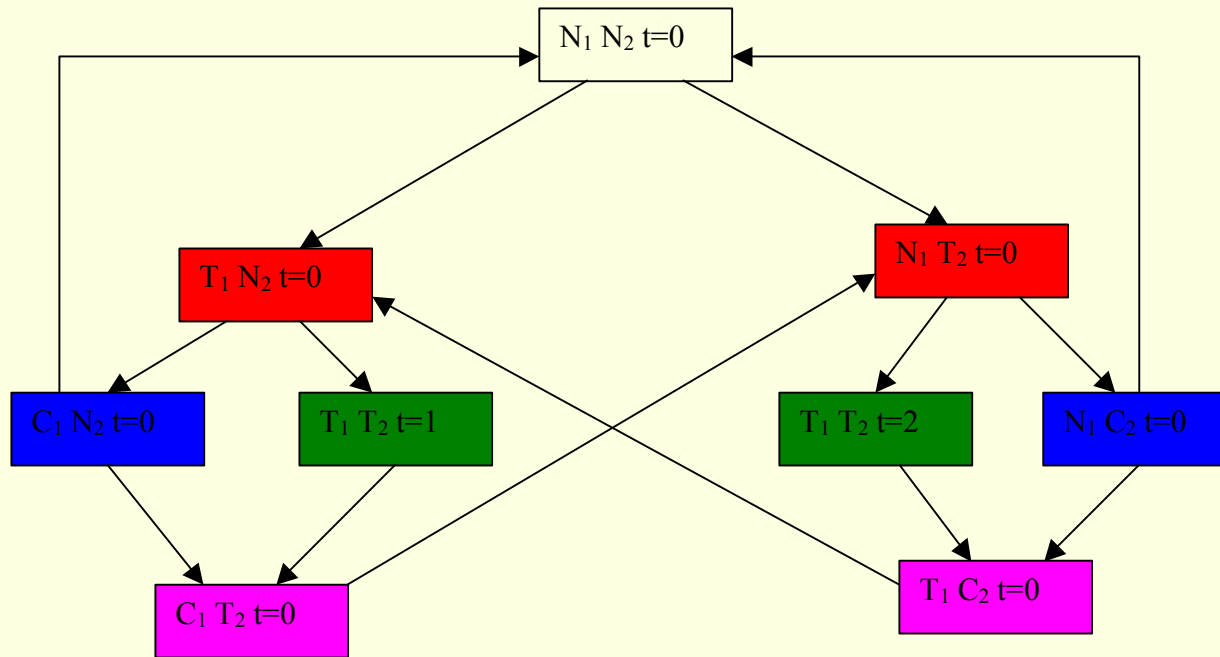
- $f = AG(EX True)$ (absence of deadlock)
- $Auto f = Sym \mathcal{I}$
- $Auto \mathcal{M} \cap Auto f =$ all circ. permutations

Example



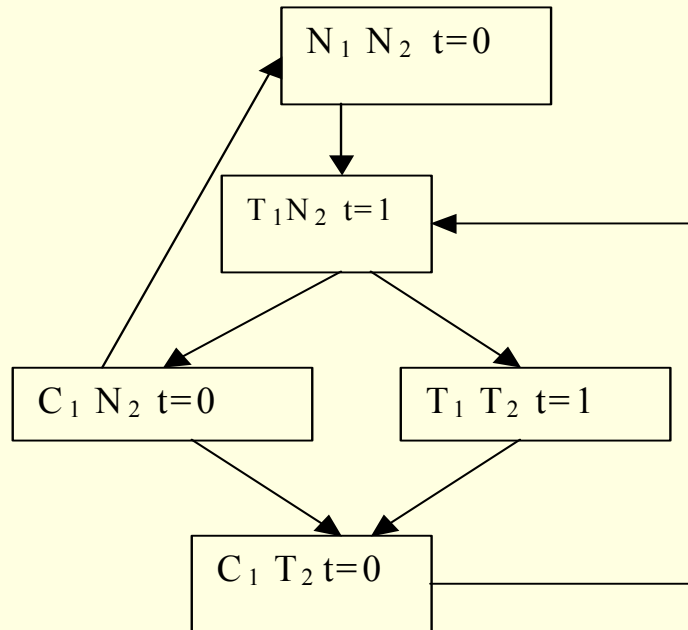
Two process Mutual Excl. $AG(\neg(C_1 \wedge C_2))$

Example contnd.



Two process Mutual Excl. $AG(\neg(C_1 \wedge C_2))$

Quotient Structure.



Fairness

- Correctness under group fairness is preserved
- $G \subseteq \text{Aut } \mathcal{M} \cap \text{Auto } f$
- Define index $i \equiv j$ iff $\exists \pi \in G$ such that $j = \pi(i)$
- C_1, C_2, \dots, C_k are the equivalence classes of indices
- Group fairness: for $\ell = 1, \dots, k$ some process $\in C_\ell$ executed infinitely often

Fairness Theorem

- (\mathcal{M}, s) satisfies f under group fairness *iff*
 $(\mathcal{M}/G, s^+)$ satisfies f under group fairness

- Example:

$f = AG ((T_1 \vee T_2) \rightarrow AF (C_1 \vee C_2))$
in the mutual exclusion example.

Incremental Computation of \mathcal{M}/G : $\mathcal{M} = (\mathcal{S}, \mathcal{R})$

$\mathcal{S}^+ = \{ s_0 \}$, s_0 -init. State

$Q = \{ s_0 \}$

While $Q \neq \text{empty}$

$s := \text{dequeue} (Q);$

 for each successor t of s

 if ($\exists u \in \mathcal{S}^+$ such that $u \equiv_G t$)

 then add (s, u) to \mathcal{R}^+

 else $\mathcal{S}^+ = \mathcal{S}^+ \cup \{t\}$,

 add (s, t) to \mathcal{R}^+

 end for.

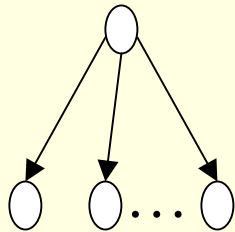
end while.

Checking $u \equiv_G t$ is a difficult problem.

Savings in the size of state space.

We can obtain exponential savings in some cases.

Resource Controller problem:



n - # of users

m - # of states of the controller

Assume each user has 3 states

\mathcal{M} has $\mathcal{O}(m \cdot 3^n)$ states.

\mathcal{M}/G has $\mathcal{O}(m \cdot n^3)$ states

Finding suitable G

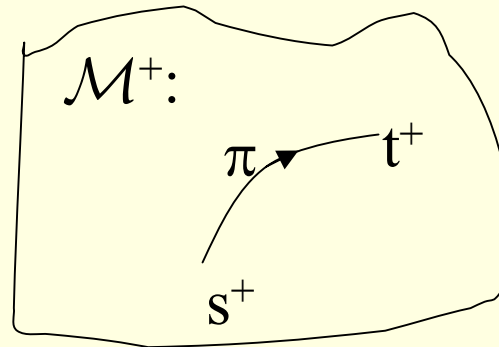
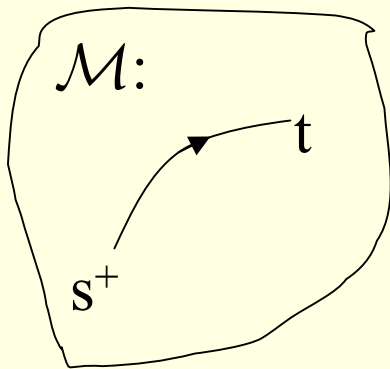
- Largest possible $G \subseteq \text{Aut } \mathcal{M} \cap \text{Auto } f$
gives maximum compression.
- Difficulties:
 - Computing $\text{Aut } \mathcal{M}$ is difficult, as hard as graph isomorphism
 - use $\text{Aut } \mathcal{K}$ (can be determined from syntax)
 - Computing $\text{Auto } f$ can be hard
 - Many times $\text{Aut } \mathcal{K}$, $\text{Auto } f$ are known in advance
 - For isomorphic processes, $\text{Aut } \mathcal{K} = \text{Aut } CG$

Automata Theoretic Approach & AQS

Don't need to consider formula symmetry.

Use an Annotated Quotient Structure. (AQS)

Take $G \subseteq \text{Aut } \mathcal{M}$



$$t = \pi(t^+)$$

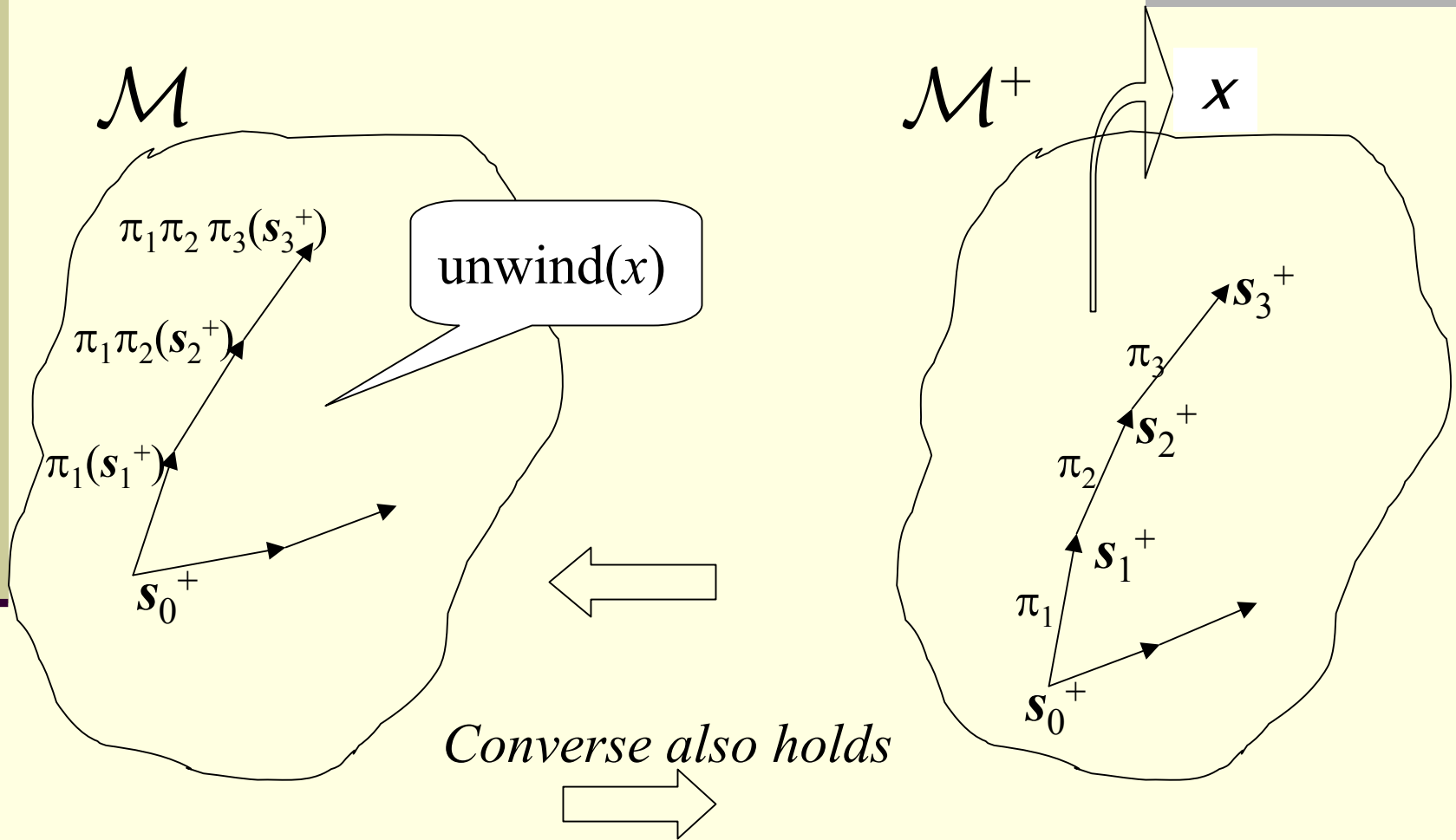
edges are labeled with permutations

if $(s^+, t) \in \mathcal{R}$ then

there is an edge from s^+ to t^+

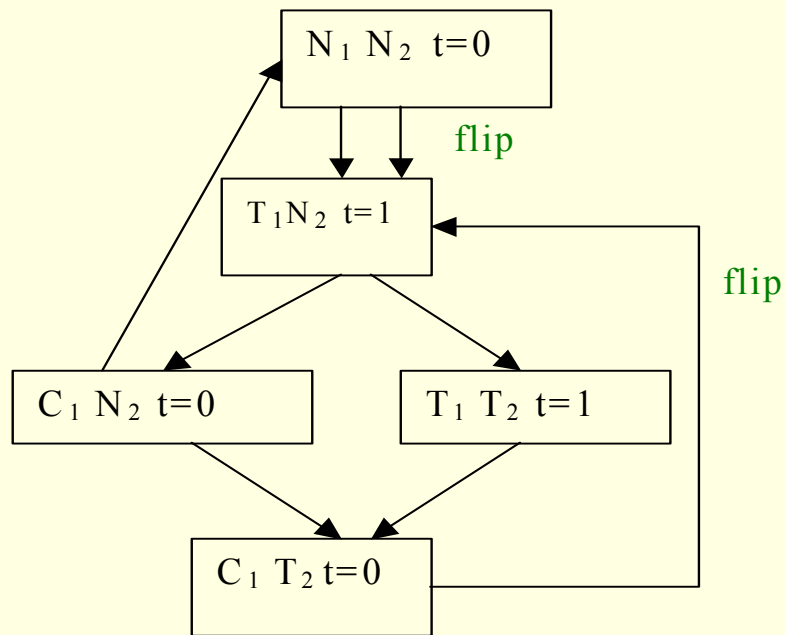
labeled with π where $t = \pi(t^+)$

Correspondence between \mathcal{M} & \mathcal{M}^+



\mathcal{M}^+ is a succinct encoding of \mathcal{M} .

Annotated Quotient:



$$G = \{\text{Id}, \text{Flip}\}$$

Correctness under fairness.

Want to check if $\mathcal{M}, s_0^+ \models E(\phi \wedge f_i)$

where ϕ expresses weak fairness. \mathcal{A} – Automaton for f .

Defn: $\mathcal{B}^+ = \mathcal{M}^+ \times \mathcal{A} \times \mathcal{I}$

$(s^+, q, i) \xrightarrow{\pi} (t^+, r, j)$ iff

$s^+ \xrightarrow{\pi} t^+ \in \mathcal{M}^+$,

$q \xrightarrow{(s^+ \downarrow i)} r \in \mathcal{A}$ and

$\pi^{-1}(i) = j$

Defn : A SCC C^+ of \mathcal{B}^+ is **green**

if $\exists (s^+, q, i) \in C^+$ such that $q \in \mathbf{GREEN}$.

Theorem:

$\mathcal{M}, s_0^+ \models E(\phi \wedge f_i)$ iff \mathcal{B}^+ contains a subtly fair and green *SCC* C^+ that is reachable from (s_0^+, q_0, i) .

How to check if C^+ is subtly fair ?

use a threaded graph C^* .

Alg to mark all states in \mathcal{M} that satisfy:

$$E(\phi \wedge f_i)$$

1. Construct \mathcal{A}
2. Construct $\mathcal{B}^+ = \mathcal{M}^+ \times \mathcal{A} \times \mathcal{I}$
3. For each *SCC* C^+ of \mathcal{B}^+
 - Check if C^+ is subtly fair
 - Construct C
 - Check if C^* is plainly fair.
4. For each $s^+ \in \mathcal{M}$
 - mark s^+ if a subtly fair and **green *SCC*** is reachable from (s^+, q_0, i) in \mathcal{B}^+

Complexity : $\mathcal{O}(|\mathcal{M}^+| \cdot |\mathcal{A}| \cdot n^2)$

Above approach can be extended to strong fairness

Complexity: $\mathcal{O}(|\mathcal{M}^+| \cdot |\mathcal{A}| \cdot n^3)$

On-the-Fly Algorithm is more subtle.

Implementation : SMC (Symmetry based Model – Checker)

- Developed at Univ. of Illinois at Chicago
- Uses AQS based approach
- Employs a variety of symmetries:
 - Program symmetry, State symmetry
- Uses variety of on-the-fly options
 - (AQS and/or product structure constructed on-the-fly)

Implementation : SMC(cont'd)

- Allows different fairness specifications
(weak/strong features)
- Used for checking real world examples,
found bugs in the *Fire-Wire* protocol.

Reduced Symmetry

- Symmetric system with asymmetric constructs.
Ex: resource controller with priorities.
- Partially Symmetric system.

Introduce

1) Guarded Quotient Structures (GQS)

Further extension of AQS

2) Two new techniques

(a) Formula decomposition

(b) Sub-formula tracking

Guarded Quotient Structures (GQS)

$G = (S, E)$ -reachability graph.

$\text{Aut}(G)$ – group of symmetries/automorphisms of G .

$\text{Aut}(G)$ – may be small. Not much compression

Add edges to G and obtain an expanded graph

$H = (S, F)$ so that

$F \supseteq E$

$\text{Aut}(H) \supseteq \text{Aut}(G)$

GQS – contd.

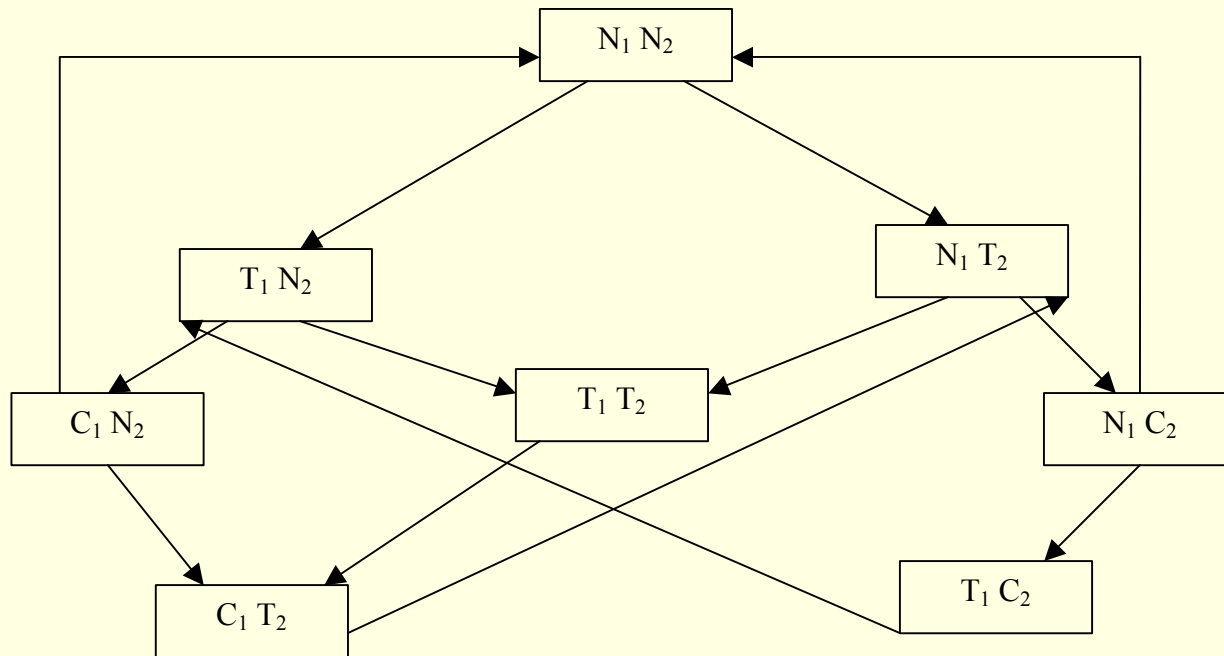
Construct the AQS of H

Add an edge condition with each edge of H (called guards)

Used during the unwinding process.

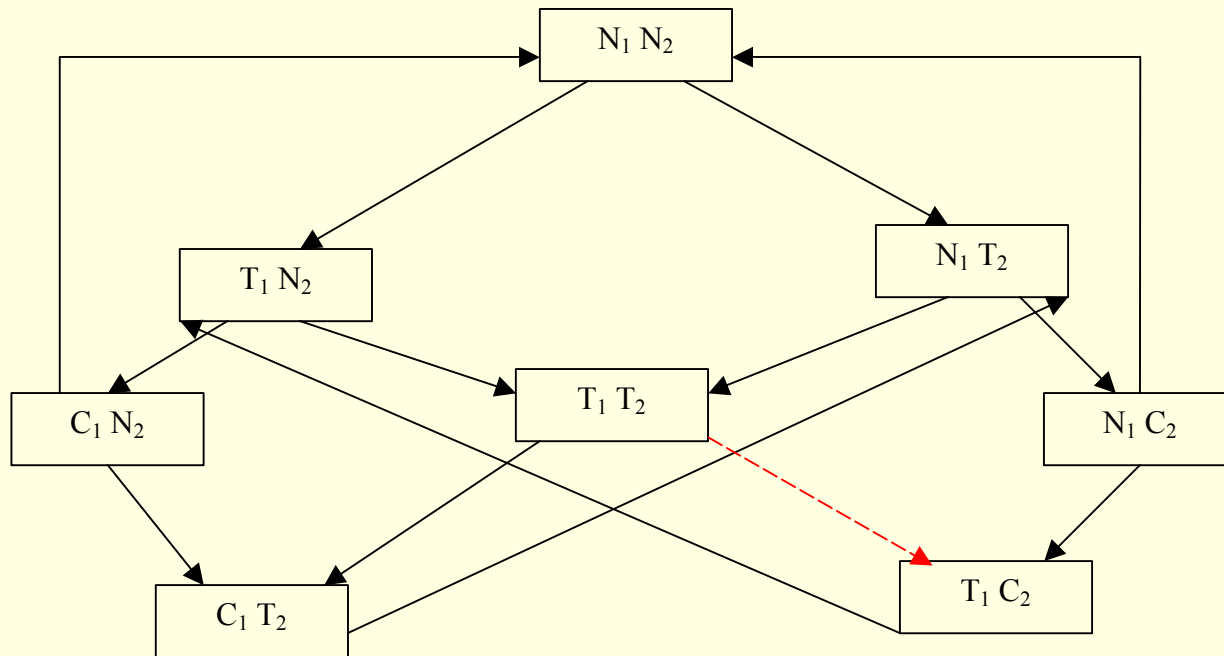
The resulting structure is GQS(G)

Example



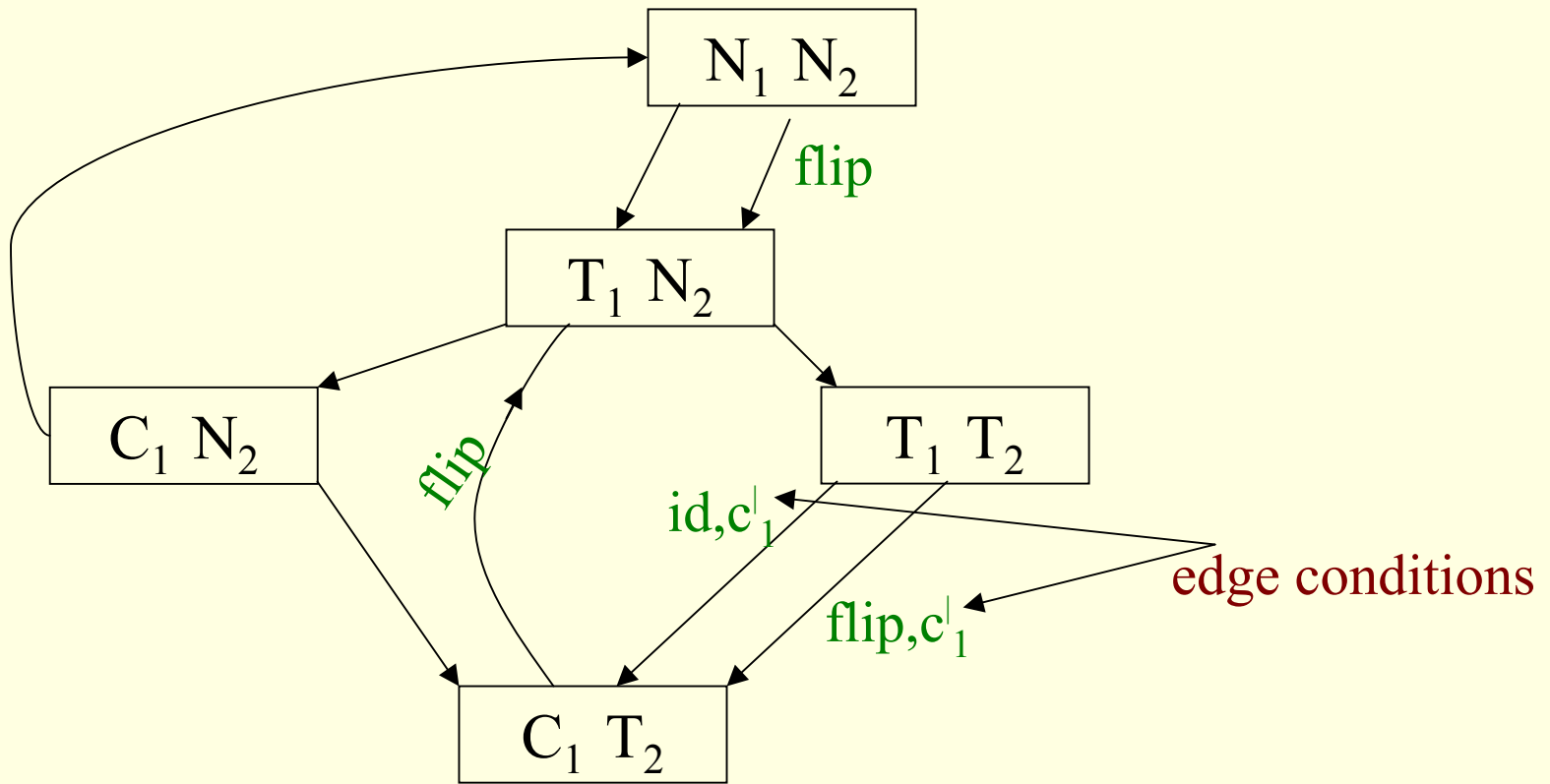
Mutual Exclusion with priority for Process 1 (The Graph G)

Example



Mutual Exclusion with priority for Process 1

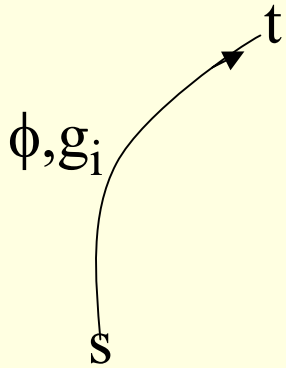
Graph H



GQS for the system with priority.

Unwinding:

GQS:



GQS-Struct

$(t, \phi^{-1}(P_1), \dots, \phi^{-1}(P_k), \phi^{-1}(\theta_1), \dots, \phi^{-1}(\theta_\ell))$

$(s, P_1, \dots, P_k, \theta_1, \dots, \theta_i, \dots, \theta_\ell)$

correspond to & track
atomic predicates in the
formula

track edge
conditions

To Check f

- Unwind the GQS to get GQS-struct
 - unwinding done w.r.t. $P_1, P_2 \dots P_k$ and the edge conditions.
 - use the edge conditions to consider only those edges in G .
 - Check f in GQS-struct
 - When optimized: GQS-struct has
 - no more nodes than as $G/\text{Aut}(G)$
 - fewer nodes in many cases
-
- Important advantage of GQS:
 - Can use formula decomposition
 - Subformula tracking

Formula Decomposition

- Suppose $f = f_1 \wedge f_2$ (f_1, f_2 are state formulas)
 - Only P_1 appears in f_1
 - Only P_2 appears in f_2
- Unwind GQS w.r.t. P_1 and check f_1
- Unwind GQS w.r.t. P_2 and check f_2
- Avoids unwinding w.r.t P_1 and P_2 simultaneously.

Generalization.

- Group Top level subformulas into classes
 - all subformulas in a class contain same atomic propositions
 - Sets of atomic propositions of different classes are disjoint.
 - Unwind GQS once for each class.

- Can achieve exponential reduction in size of state space.

Subformula Tracking

Unwind the GQS w.r.t. non atomic state subformulas

To check f

- Choose a good maximal independent set $R = \{R_1, R_2, \dots, R_k\}$ of state subformulas of f
- Replace each R_i in f by a new atomic proposition r_i and obtain \bar{f}
- Unwind GQS w.r.t. R to obtain GQS-struct
 - recursively determine which states of the GQS satisfy the subformulas in R
 - label a node in GQS-struct with r_i if the corresponding GQS node satisfies R_i (for $i = 1, \dots, k$)
- Check if \bar{f} is satisfied in GQS-struct

Example

Consider a Resource Controller where process 0 is the controller and others are user processes

(i.e. processes 1,2,...n)

$$f = E(P_1 \mathcal{U} g)$$

where $\{ g = \bigwedge_{2 \leq i \leq n} E(h(i)) \}$

$$R = \{ P_1, g \}$$

Unwind GQS w.r.t. R to obtain GQS-struct

Recursively determine states that satisfy g (use formula decomposition)

Check \overline{f} in GQS-struct

-
- In general, subformula tracking and formula decomposition are used recursively.
 - Good independent sets have symmetric or partially symmetric subformulas.

Implementation

- Extended SMC to PSMC
 - priorities can be specified in PSMC
- Used GQS together with
 - formula decomposition and
 - sub –formula tracking
- Checked *Fire-Wire* Protocol with priorities.

Conclusions.

- Symmetry reductions help in tackling the state explosion.
- Can also be used for systems with less symmetry.
- Have been implemented and used for verifying real-life protocols.

▪ Future Work.

- Need to effectively combine with other methods.
- Applications to software verification need to be explored.

Further Details @Home Page:

www.cs.uic.edu/~sistla