

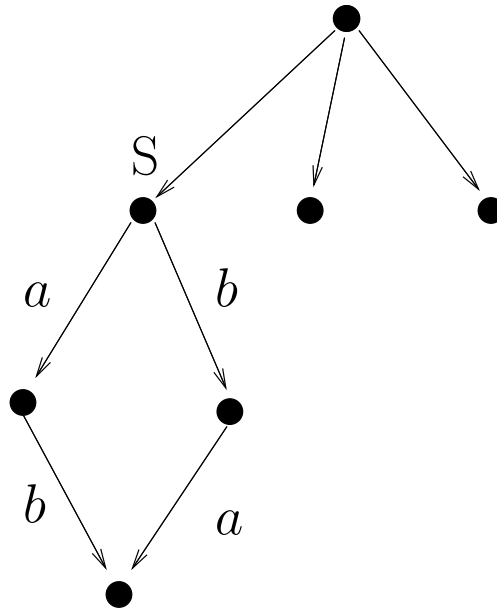
Part 4

Symbolic Partial Order Reduction

Joint work with Ritwik Bhattacharya and Ganesh Gopalakrishnan,
University of Utah

Partial Order Reduction

Partial Order Reduction uses independence to reduce the state space for model checking, while still detecting all errors. [Clarke, Grumberg, Peled 1999]



- Full search examines each enabled transition at each state.
- Partial Order Reduction examines an *ample set* of transitions at each state.
- All linear time temporal properties not involving next-time are preserved

Definitions about Transitions

For a transition a we write

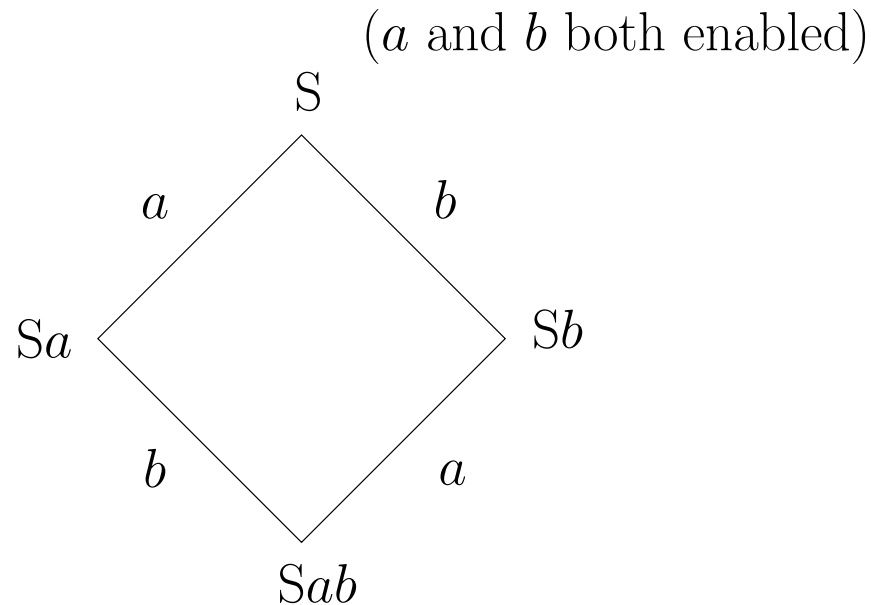
$$en_a(S) : States \rightarrow Boolean$$

for the enabling condition at state S , and

$$a(S) : States \rightarrow States$$

for the successor state from state S by transition a .

Independent Transitions



Transitions a, b are *independent* iff for all states S :

- $en_a(S) \wedge en_b(S) \Rightarrow en_a(b(S)) \wedge en_b(a(S))$
- $en_a(S) \wedge en_b(S) \Rightarrow a(b(S)) = b(a(S))$

A traditional implementation uses *variable occurrence* to check independence.

This approach is not accurate enough for protocols having complex transition rules.

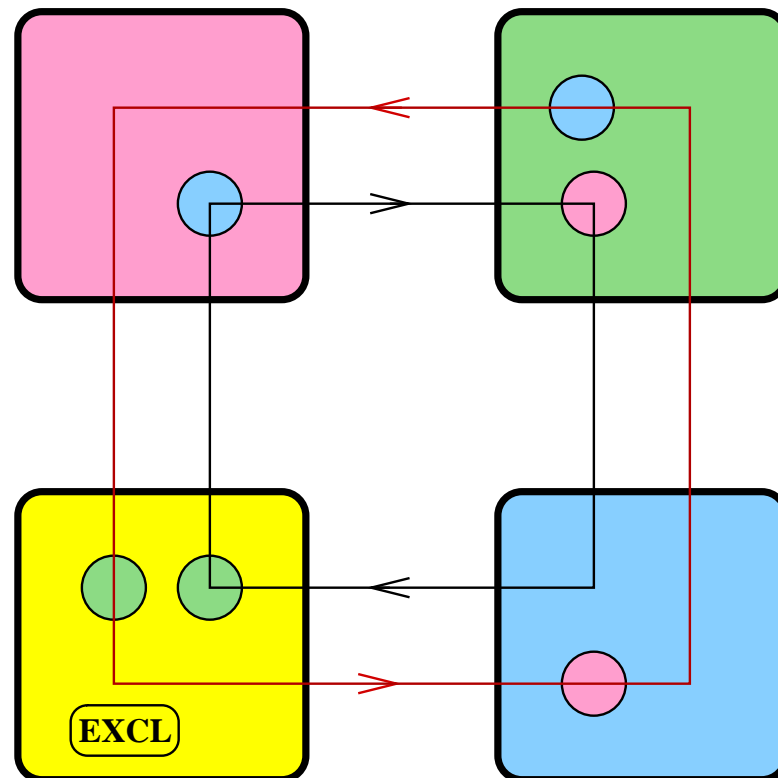
Symbolic Partial Order Reduction

- We are investigating the use of symbolic simulation and propositional reasoning to implement a more accurate check for independence.
- Some initial results are presented in a technical report

A Symbolic Partial Order Reduction Algorithm for Rule Based Transition Systems,
R. Bhattacharya, S. German, G. Goplakrishnan
UUCS-03-028, School of Computing, University of Utah, December 2003

An Example

- Token-passing mutual exclusion protocol
- Two concentric rings
- Example has characteristics of a cache protocol.



Protocol Rules

- If a node does not have the token, it can issue a request by sending two messages on the ring.
- Requests are either granted or rejected.
- If a granted request returns to the issuing node, the node enters the exclusive state.
- When both messages have returned to issuing node, if the node has received a grant, it will grant requests from other nodes.

Message Status

A message can be in one of the following states:

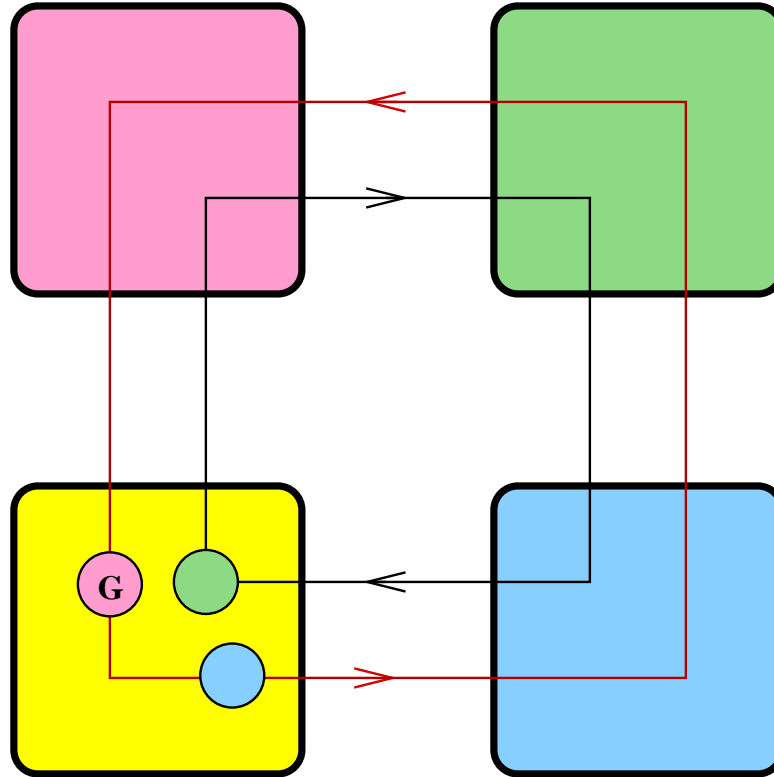
- Request
- Reject
- Grant

Rules for Granting and Rejecting

1. If a Request arrives at a node in the Granting state, the request is Granted.
2. If a Request arrives at a node that has the Exclusive token but is not ready to Grant, the request is Rejected.
3. If a Request arrives at a node that contains a Grant message for a different owner, the Request is Rejected.
4. If a Request arrives at a node containing a Rejected Request from the same owner, the Request is Rejected.

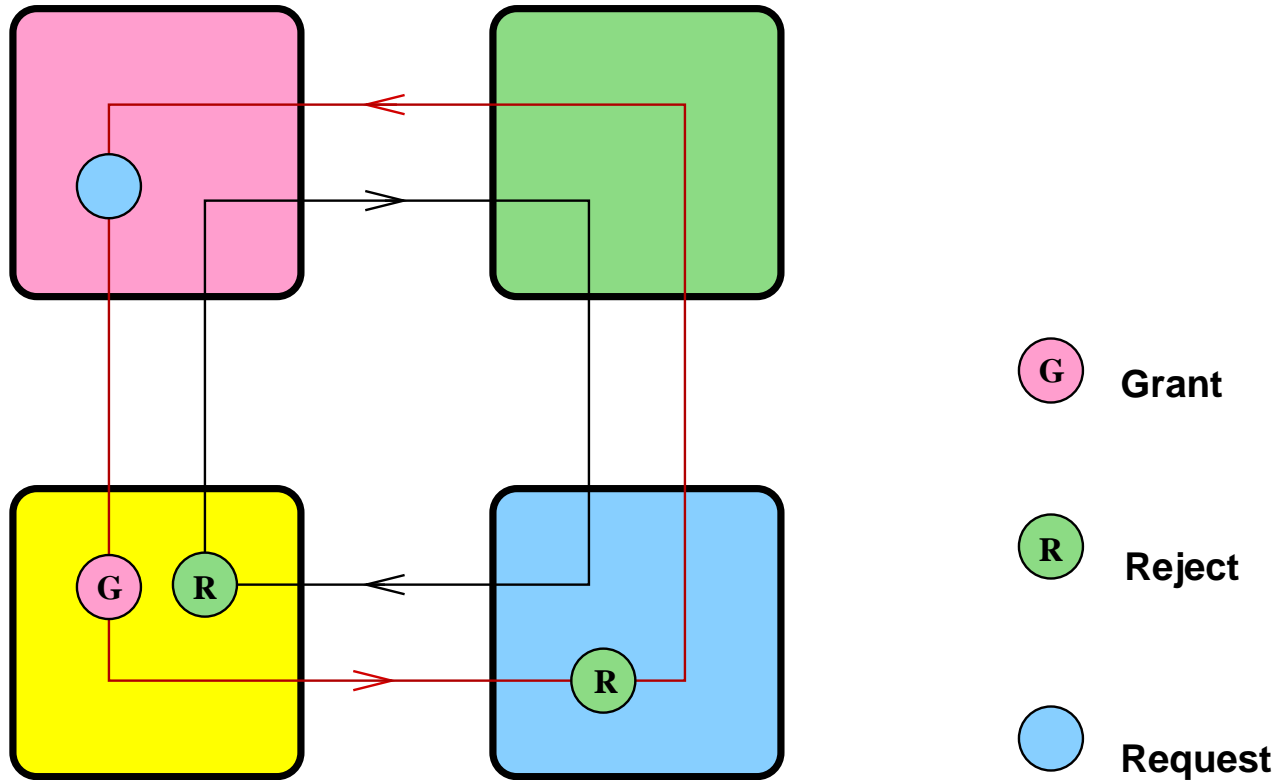
Message Priority

- A message with status Grant has priority over other messages for transfer out of the node.



Acceleration of Rejected Messages

- Rejected messages have little interaction with system.
- Intuitively, rejected messages can be accelerated.



Acceleration of rejected messages gives 5 to 6 times reduction of state space.

Using Deduction to Justify an Ample Set

If a transition t is independent of all other transitions, if t is *invisible* and if we observe a *cycle condition*, then we can use $\{t\}$ as a ample set.

We would like to show that transfer of Rejected Requests satisfies these conditions

- Independence is non-trivial because of message priority.
 - Use symbolic checking for independence.
- Independence may only be true in the reachable state space.
 - Prove invariants of the transition system to justify independence.
- It may be necessary to *split* a transition into smaller cases which are independent.

```
guard ==> if b then S1 else S2 endif
```

```
guard & b ==> S1
```

```
guard & !b ==> S2
```

Appendix B. Mutual Exclusion Protocol Model

```
/* ring.m == tutorial example of a ring fabric for a mutual exclusion
    protocol, showing a manual partial order reduction

    S.M. German    19 August 2004

*/

const node_count: 4;

const reduce: false; -- set this constant to true to apply PO reduction

type
  node_id: 0 .. node_count - 1;

  message_id: 0 .. (2 * node_count) - 1;

  active_count: 0 .. node_count - 1;

  status_type: enum{inactive, request, reject, grant};

  message_type:
    record
      loc: node_id;
      status: status_type;
    endrecord;

  node_type:
    record
      active: active_count;
```

```

        exclusive: boolean;
        requesting: boolean;
    endrecord;

var
    node: array[node_id] of node_type;
    msg: array[message_id] of message_type;

function left(n:node_id): message_id;
begin
    return 2 * n;
end;

function right(n:node_id): message_id;
begin
    return (2 * n) + 1;
end;

function left_succ(n: node_id): node_id;
begin
    return (n + 1) % node_count;
end;

function right_succ(n: node_id): node_id;
begin
    return (n + node_count - 1) % node_count;
end;

function next_node(m: message_id): node_id;
begin
    alias n: msg[m].loc do
        if m % 2 = 0 then return left_succ(n) else return right_succ(n); endif;
    endalias;
end;

```

```

function other_message(m: message_id): message_id;
begin
  if m % 2 = 0 then return m + 1 else return m -1; endif;
end;

function home_node(m: message_id): node_id;
begin
  return m / 2;
end;

function message_priority(m: message_id): boolean;
begin
  alias ms: msg[m] do
  return ms.status = grant
    | !exists mm: message_id do
      msg[mm].loc = ms.loc
      & msg[mm].status = grant
      endexists;
endalias; end;

procedure transfer(m: message_id);
var loc: node_id;
begin
  loc := next_node(m);
  msg[m].loc := loc;

  if loc = home_node(m)
  then
    if msg[m].status = grant then node[loc].exclusive := true; endif;
    clear msg[m];
    if msg[other_message(m)].status = inactive then
      node[loc].requesting := false;
    endif;

  elsif msg[m].status = request

```

```

    & node[loc].exclusive
-- 16 Aug 04
-- loc only grants when both of its messages have returned
    & !node[loc].requesting
  then msg[m].status := grant;
       node[loc].exclusive := false;

-- 16 Aug 04
-- loc rejects a request if it has the exclusive state but it still has
-- a request message outstanding
  elseif msg[m].status = request
    & (( exists mm: message_id do
        msg[mm].loc = loc
        & home_node(m) != home_node(mm)
        & msg[mm].status = grant
        endexists )
      | node[loc].exclusive )

  then msg[m].status := reject;
  endif;

-- this part is optional:
-- if both messages from a given home are located at the same node on
-- the ring, and one of the messages is rejected, mark the other
-- message as rejected.
  if msg[other_message(m)].loc = loc
    & ((msg[m].status = reject & msg[other_message(m)].status = request)
      | (msg[m].status = request & msg[other_message(m)].status = reject))
  then msg[m].status := reject;
       msg[other_message(m)].status := reject;
  endif;
end;

startstate
  clear node;

```

```

clear msg;
node[0].exclusive := true;
end;

ruleset n: node_id do
  rule "node issues request"
    !node[n].requesting
& !node[n].exclusive ==>
begin
  msg[left(n)].status := request;
  msg[right(n)].status := request;
  msg[left(n)].loc := n;
  msg[right(n)].loc := n;
  node[n].requesting := true;
end; endruleset;

-- unreduced form of rule
ruleset m: message_id do
  rule "transfer message"
    !reduce
& msg[m].status != inactive
& message_priority(m) ==>
  transfer(m)
endrule; endruleset;

ruleset m: message_id do
  rule "transfer unrejected message"
    reduce
& msg[m].status != inactive
& message_priority(m)
& !exists o: message_id do
  msg[o].status = reject
  & message_priority(o)
endexists
==>

```

```

    transfer(m)
endrule; endruleset;

ruleset m: message_id do
  rule "transfer rejected message"
    reduce
& msg[m].status = reject
& message_priority(m)
& !exists o: message_id do
  o < m
  & msg[o].status = reject
  & message_priority(o)
endexists
==>
  transfer(m)
endrule; endruleset;

invariant "excl" forall i: node_id; j: node_id do
  ! (i != j & node[i].exclusive & node[j].exclusive) endforall;

invariant "exclusive exists"
  exists n: node_id do node[n].exclusive endexists
| exists m: message_id do msg[m].status = grant endexists;

/* PO reduction data:
8/16 model change: grant is not allowed if request is still active

4 nodes:  before 8/16 change, unreduced 30.1M, reduced 5.2M
           30.1 / 5.2 = 5.8

           with 8/16 change, unreduced 20.4M, reduced 3.9M
           20.4 / 3.9 = 5.2
*/

```