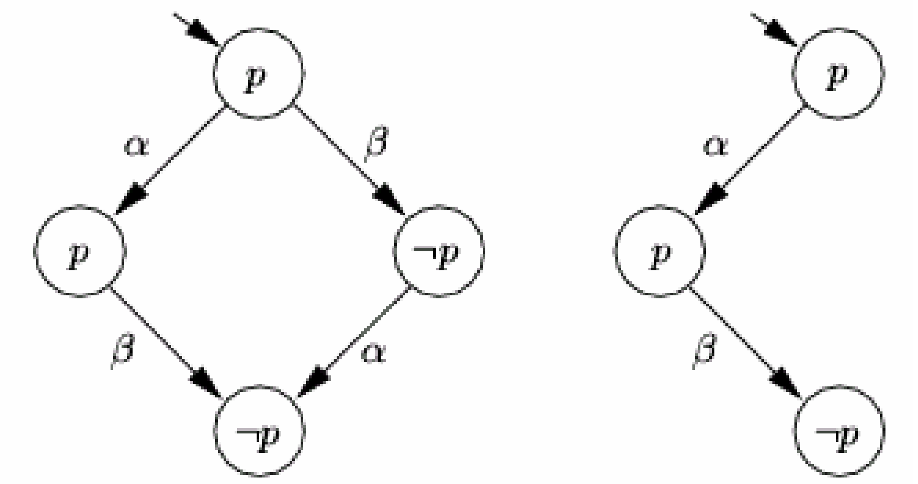
Partial Order Reduction

Part 8

Idea

- **State Space Reduction**
  - reduce the number of explored states and transitions by exploiting redundancies in the state space
    - redundancies with respect to a property
  - requirement on the reduction
    - the property holds in the reduced state space iff it holds in the full state space
- **Example**
  - exploit commutativity of concurrently enabled transitions
  - property: $\Diamond\neg p$



Concrete and Abstracted State Spaces

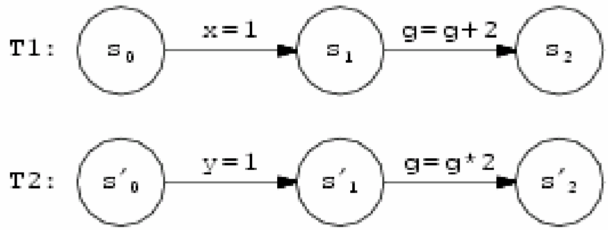


Figure 4.2 The Labeled Transition Systems T1 and T2

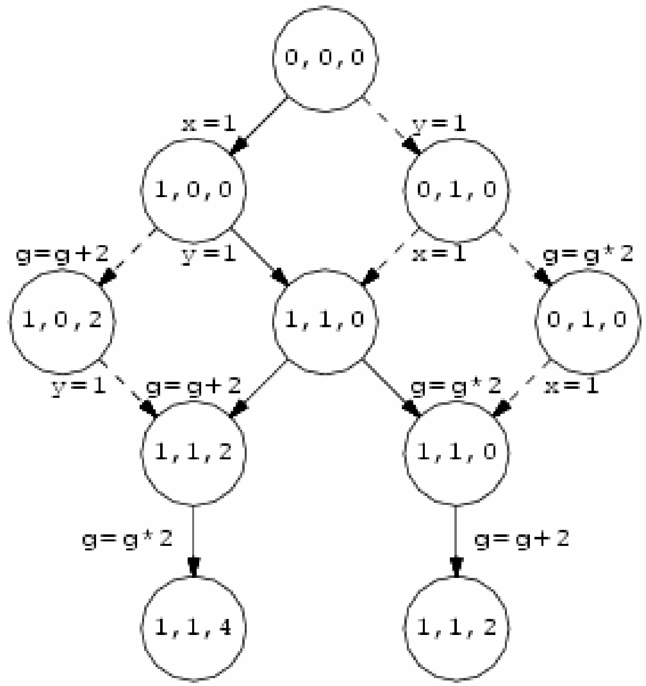
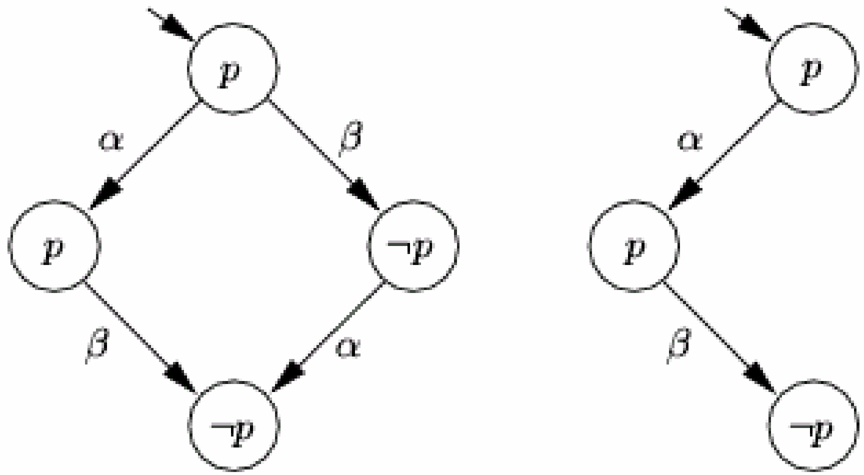


Figure 4.3 Full and Reduced Depth-First Search for T1×T2

state labels: (x, y, g), reduced state graph = solid line arrows

- Properties valid in full and reduced state graph
  - $\Box(g=0 \vee g>x)$
  - $\Diamond\ (g \geq 2)$
  - $(g=0)\ U\ (x=1)$
- Property valid only in reduced state graph
  - $\Box(x \geq y)$

Relevance of Order of Concurrent Transitions



- **Note**
  - order in which concurrent transitions are executed is often irrelevant with respect to the overall behaviour of the system
  - yet, LTL permits to discriminate between otherwise equivalent sequences
    - $p \wedge \bigcirc p \wedge \Diamond \neg p$
  - take advantage of irrelevance of concurrent transitions when ordering doesn't matter for property to be checked

Depth First Search

- **On-the-fly DFS with Partial Order Reduction**

```
procedure explore_statespace
        set on_stack(q0);
        dfs(q0);
end procedure

procedure dfs(q)
        work_set(q) := ample(q)
        local q';
        hash(q);
        for all elements q' of work_set(q) do
                if q' not in hashtable
                        then set on_stack(q');
                                dfs(q');
                end if;
        end do;
        set completed(q);
end procedure
```

Ample Set

- **Calculation of ample(q) $\subseteq$ enabled(q)**
  - include sufficiently many elements from enabled(q) so that model checking algorithm delivers correct results
  - use of ample(q) should lead to a significantly smaller state graph (in terms of states and transitions)
  - computation of ample(q) should be doable with acceptable computation overhead

State Transition System

- **State Transition System**
  - let AP a set of atomic propositions
  - a state transition system is a tuple (S, T, S_0, L) where
    - S: finite set of states
    - $S_0 \subseteq S$: finite set of inital states
    - $L: S \to 2^{AP}$: function that labels every state with the atomic propositions true in that state
    - T: finite set of transition relations so that for each $\alpha \in T$, $\alpha \subseteq S \times S$
  - let $\alpha \in T$, $s \in S$, then
    - $\alpha \in enabled(s)$ iff $(\exists s' \in S)((s, s') \in \alpha)$
    - α is deterministic if for every s, there is at most one s' so that $(s, s') \in \alpha$
      - henceforth we only consider deterministic transitions
    - we write $s'=\alpha(s)$ for $(s, s') \in \alpha$
  - a path from a state s is a finite or infinite sequence

    $\pi = s_0 \rightarrow_{\alpha 0} s_1 \rightarrow_{\alpha 1} \ldots$

    so that $s = s_0$ and for all i, $(s_i, s_{i+1}) \in \alpha_i$

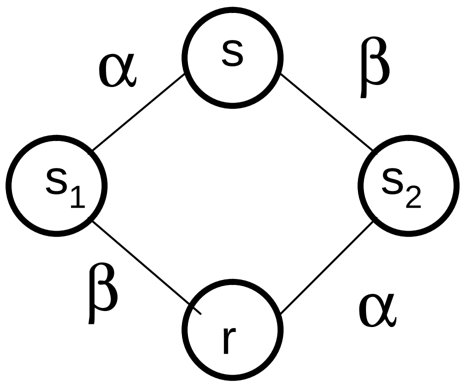
Independence

- **Independence of transitions**
  - $I \subseteq T \times T$ is an independence relation if I is symmetric and antireflexive and the following conditions hold for each $s \in S$ and for each $(\alpha, \beta) \in I$:
    1. if $\alpha, \beta \in$ enabled(s) then $\alpha \in$ enabled(β(s))

       enabledness: a pair of independent transtions do not enable each other when taken
    2. if $\alpha, \beta \in$ enabled(s) then $\alpha(\beta(s)) = \beta(\alpha(s))$

       communtativity: executing pair of independent transitions in any order results in same sate
  - $D := (T \times T) - I$ (the dependence relation)

Correctness of Pruning



- **Elemination of one of the branches**
  - may deliver incorrect results if
    1. s_1 and s_2 may influence the outcome of the property check, i.e., the property is not insensitive to whether s_1 or s_2 is being reached
    2. s_1 or s_2 may have successor states other than r that would be pruned in case either of the two states did not belong to the reduced state space

Invisibility

- **Invisibility**
  - let $\mathcal{T}$=(S, T, S_0, L) a state transition system
  - let AP' $\subseteq$ AP
  - $\alpha \in$ T is invisible wrt. AP' if

    $(\forall\ s, s' \in S \mid s' = \alpha(s))\ ((L(s) \cap AP') = (L(s') \cap AP'))$

    (i.e., a transition is invisible with respect to some selected set of propositions if its execution dosn't change the truth value for the selected set of propositions.)

Invariance under Stuttering

- **Invariance under Stuttering**
  - relationship between identically labeled sequences of states along the path through a state transition system
  - two path σ and ρ through a state transition system are stuttering equivalent (written as $\sigma \sim_{st} \rho$) if the following condition holds:
    - there are two infinite sequences of integers

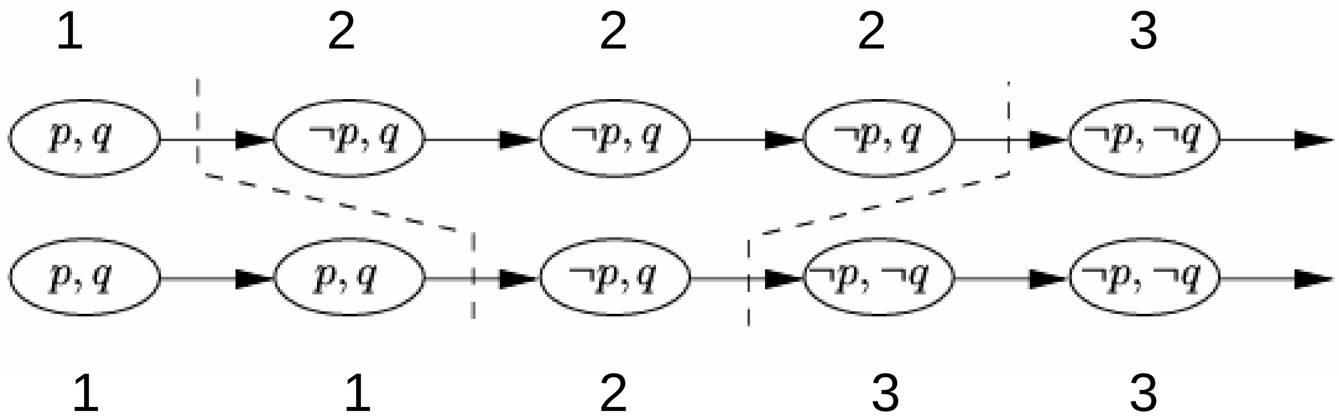
      $0 = i_0 < i_1 < \ldots$

      $0 = j_0 < j_1 < \ldots$

      such that for every $k \geq 0$

      $$L(s_{i_k}) = L(s_{i_k+1}) = L(s_{i_{k+1}-1}) = L(r_{j_k}) = L(r_{j_k+1}) = L(r_{j_{k+1}-1})$$

      where all s_i are states from σ and all r_i are states from ρ.
    - identically labeled sequences of states are called blocks



Invariance under Stuttering

- **Invariance under Stuttering for LTL fomulae**
  - an LTL formula f is invariant under stuttering if
    - for each pair of paths π and π' such that $\pi \sim_{st} \pi'$

      $\pi \models f$ iff $\pi' \models f$
- **Theorem**
  - let LTL_χ denote the set of all LTL formulae free of the nexttime ($\bigcirc$) operator
  - any property expressible in LTL_χ is invariant under stuttering
  - proof: by simple structural induction over the length of LTL_χ formulae

Invariance under Stuttering

Stutter Invariance for Transition Systems

- Let M and M' state transition systems. M and M' are stutter invariant iff
  - they have the same set of initial states
  - for each path σ of M that starts in an initial state of M there is a path σ' of M' that starts in an initial state of M' such that $\sigma \sim_{st} \sigma'$
  - for each path σ' of M' that starts in an initial state of M' there is a path σ of M that starts in an initial state of M such that $\sigma' \sim_{st} \sigma$
- Theorem
  - Let M and M' two stuttering equivalent state transition systems. Then, for every property expressed by an LTL_χ formula f and every initial state $s \in S_0$ the following holds true

    $(M, s) \models f$ iff $(M', s) \models f$
  - I.e., LTL_χ formulae cannot distinguish between stuttering equivalent state transition systems

Partial Order Reduction for LTL_χ

- **Reduction**
  - exploit communtativity and invisibility for stutter invariant property specifications to reduce the size of the explored state space
  - now: conditions for the construction of ample set
    - s is fully expanded, iff enabled(s) = ample(s)
    - otherwise, provide conditions for selecting ample(s) such that reduced state space satisfies property expressed by LTL_χ formula f

      C0: at-least-one-successor rule

      C1: dependent-transition rule

      C2: invisibility rule

      C3: cycle condition
    - for LTL_χ property f, reduction depends on the set AP_f of atomic propositions occurring in f

Partial Order Reduction for LTL_c

- **C0: At-Least-One-Successor Rule**
  - $(\forall\ s \in S)(\text{ample}(s)=\emptyset \text{ iff enabled}(s)=\emptyset)$
    - i.e., if a state has at least one successor in the full state space, it has at least one successor in the reduced state space

Partial Order Reduction for LTL_c

- **C1: Dependent-Transition Rule**
  - for all states $s \in S$
    - for all paths in the full state space starting at s, the following holds true:
      - a transition α' that is dependent on a transition $\alpha \in$ ample(s) cannot be executed without a transition from ample(s) occurring first
  - Theorem
    - the transitions in enabled(s) – ample(s) are all independent of those in ample(s)
    - Poof: Let $\gamma \in$ enabled(s) – ample(s). Let $(\gamma, \delta) \in D$ with $\delta \in$ ample(s). Since $\gamma \in$ enabled(s), in the full state graph there is a path starting with γ. Then, a transition dependent on some other transition in ample(s) would be executed before transition in ample(s), contradicting C1.

Partial Order Reduction for LTL_c

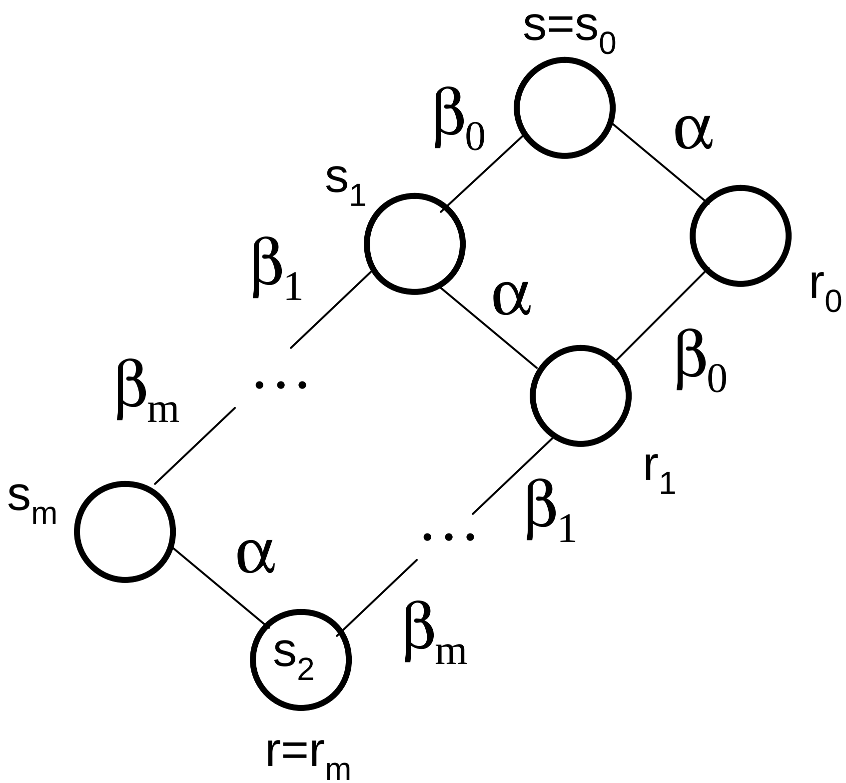
- **Preservation of Correctness when Pruning Graph**
  - ensure that reduced DFS algorithm, when choosing the next transition to explore from ample(s), does not prune any parts of the graph that are essential to the property
  - C1 implies such a graph will have any of the following two forms:
    - case 1: $\beta_0\beta_1\ldots\beta_m\alpha$, where $\alpha \in$ ample(s) and each β_i is independent of all transitions in ample(s), including α, or
    - case 2: $\beta_0\beta_1\ldots$ where β_i is independent of all transitions in ample(s)
  - C1 also implies that
    - if along a finite sequence of transitions $\beta_0\beta_1\ldots\beta_m$ starting from s none of the transitions of ample(s) have occurred, then all transitons in ample(s) remain enabled in all states reached by this transition sequence

Partial Order Reduction for LTL_c

- **Preservation of Correctness when Pruning Graph**
  - case 1: $\beta_0\beta_1\ldots\beta_m\alpha$
    - assume $\beta_0\beta_1\ldots\beta_m\alpha$ leads to a state r and $\beta_0\beta_1\ldots\beta_m\alpha$ is pruned
    - due to enabledness and commutativity applied m times, we can construct $\alpha\beta_0\beta_1\ldots\beta_m$ that also leads from s into state r
  - case 2: $\beta_0\beta_1\ldots$ c.f. later

Partial Order Reduction for LTL_c

- **Preservation of Correctness when Pruning Graph**
  - Pruning of *states* based on independence of *transitions*



  - $\sigma = s_0 s_1 \ldots s_m r$ can only be pruned if it is stuttering equivalent to $\rho = s r_0 r_1 \ldots r_m$
    - property must be unable to distinguish between σ and ρ
    - this is the case if α is invisible, i.e.,

      $L(s_i) = L(r_i)$ for all $0 \leq i \leq m$

Partial Order Reduction for LTL_c

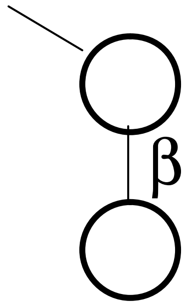
- **C2: Invisibility Rule**
  - for all states $s \in S$
    - if s is not fully expanded, then every $\alpha \in$ ample(s) is invisible

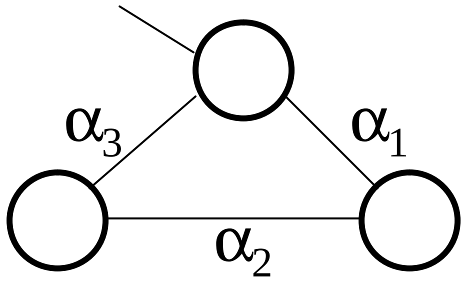
Partial Order Reduction for LTL_c

- **Preservation of Correctness when Pruning Graph**
  - case 2: $\beta_0\beta_1\ldots$
    - $\beta_0\beta_1\ldots$does not include any transition from ample(s)
    - due to C2, all transitions in ample(s) are invisible
    - let $\alpha \in$ ample(s)
    - then $\alpha\beta_0\beta_1\ldots$is stuttering equivalent to $\beta_0\beta_1\ldots$
    - i.e., even though $\beta_0\beta_1\ldots$is not included in the reduced state graph, a stuttering equivalent path $\alpha\beta_0\beta_1\ldots$is included
      - since the property cannot distinguish between both transition sequences, the pruning of $\beta_0\beta_1\ldots$does not matter

Partial Order Reduction for LTL_c

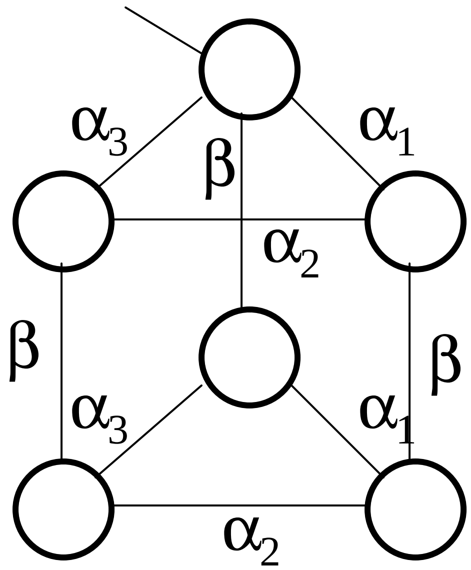
- **Preservation of Correctness when Pruning Graph**
  - problem: transitions may get deferred forever, because of cycle in constructed state graph





2 concurrent processes

- β independent of α_1, α_2 and α_3
- α_1, α_2 and α_3 are interdependent and invisible
- β is visible

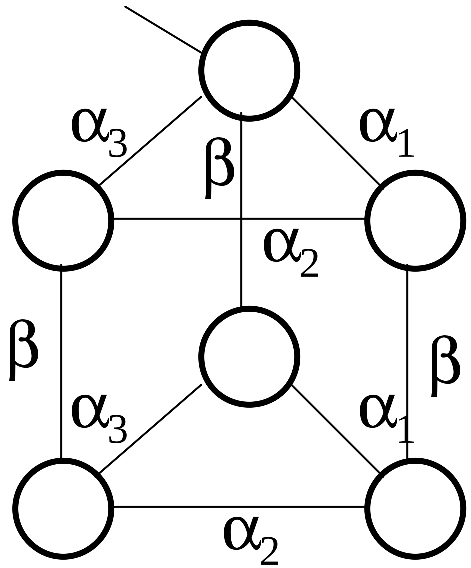


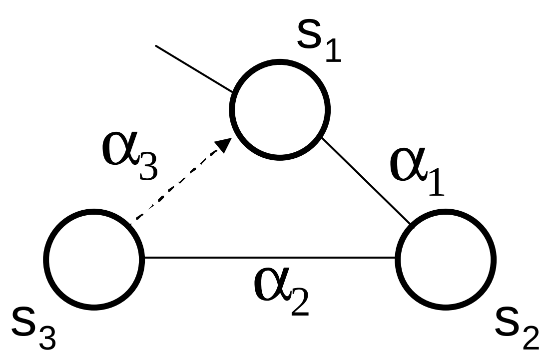
full state graph of composed system

Partial Order Reduction for LTL_c

♦ Preservation of Correctness when Pruning Graph

- construction of reduced state graph
  - initial state s_1: ample(s_1) := $\{\alpha_1\}$, satisfies C0, C1, C2
  - $s_2 := \alpha_1(s_1)$, ample(s_2) := $\{\alpha_2\}$, satisfies C0, C1, C2
  - $s_3 := \alpha_2(s_2)$, ample(s_3) := $\{\alpha_3\}$, satisfies C0, C1, C2
- problem
  - cycle s_1 s_2, s_3, s_1 does not execute visible transition β
    - β defered indefinitely

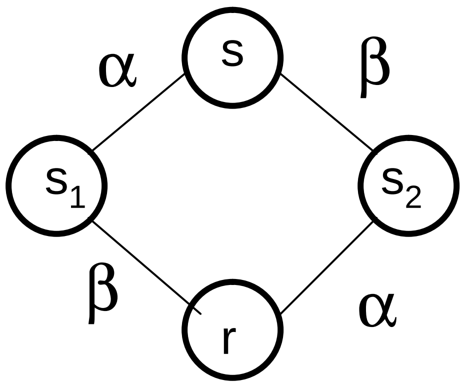




Partial Order Reduction for LTL_c

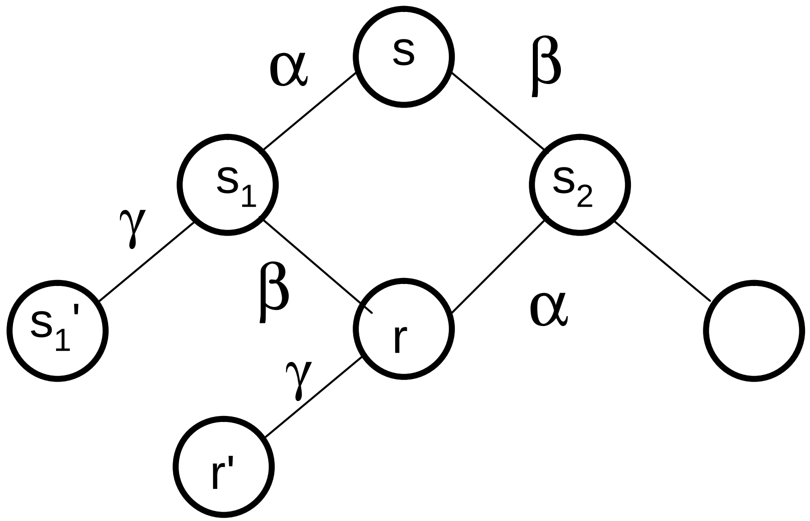
- **C3: Cycle Condition**
  - the reduced state graph may not contain a cycle in which $\alpha \in$ enabled(s) for some state s of the cycle so that $\alpha \notin$ ample(s') for all states s'=s of the cycle

Correctness of Pruning



- **Elemination of one of the branches**
  - may deliver incorrect results if
    1. s_1 and s_2 may influence the outcome of the property check, i.e., the property is not insensitive to whether s_1 or s_2 is being reached
    - by C2, β must be invisible
    - consequently, s,s_1,r and s,s_2,r are stuttering equivalent
    - no LTL_χ property capable of discriminating between these sequences

Correctness of Pruning



- **Elemination of one of the branches**
  - may deliver incorrect results if
    2. s_1 or s_2 may have successor states other than r that would be pruned in case either of the two states did not belong to the reduced state space
    - show that γ is still enabled in r and that α,γ and β,α,γ correspond to stuttering equivalent state sequences
      - γ and β are independent (C1)
      - therefore γ enabled in r
      - assume γ leads to r' from r and to s_1' from s_1
      - since β invisible, s,s_1,s' and s,s_2,r,r' are stuttering invariant

Computing Ample Sets

- **C1**
  - Theorem
    - checking condition C1 for some state s and a set T $\subseteq$ enabled(s) is at least as hard as checking reachability for the full state space
    - proof (c.f. [Clarke, Grumberg and Peled] 10.5.1)
      - basically, you need to traverse all successor states to a state in which some transition is enabled to ensure ordering constraint
  - solution: use of an approximating heuristics

Computing Ample Sets

- **C3**
  - refers to complete reduced state graph, but cycle checking is important
  - desire to compute C3 on-the-fly
  - Theorem
    - a sufficient condition for C3 is that at least one state along each cycle in the reduced state graph is fully expanded
    - proof (c.f. [Clarke, Grumberg and Peled] 10.5.1)
  - for depth-first search, this can be computed on-the fly
- **C3'**
  - for all states s, if s is not fully expanded, then no transition in ample(s) may reach a state that is on the search stack
  - over-approximates C3, i.e., this is a stronger condition than C3
    - leads potentially to less reduction

Bibliographic References

- [Clarke, Grumberg and Peled] E. Clarke, O. Grumberg and D. Peled, *Model Checking*, MIT Press, Cambridge, 1999, Chapter 10.
- [Holzmann 95] G. Holzmann, *The Verification of Concurrent Systems*, unpublished manuscript, AT&T Inc., 1995, Chapter 4.