Path Analysis

Ulrich Scholz
TU Darmstadt

November 5th, 2001

Path Analysis

Ulrich Scholz
TU Darmstadt

November 5th, 2001

“work in progress”

Introduction: Preprocessing

Transformation of a problem into an easier one by

- reducing the search space
- making knowledge explicit

Preprocessing $\neq$ Trying to solve the problem

Desired properties:

- fast runtime (low polynomial of problem size)
- optimality preserving

Introduction: Path Analysis

Preprocessing technique

Reduces state stransition graph by identifying irrelevant transitions

Based on finite state machines (FSMs)
as substructures of the state space

First idea of using PA for solving a search problem:

- Find relevant solutions for FSMs
- Combine them to a solution to search problem

Problem: FSMs are not independent of each other

Outline

- State space and FSMs
- How to identify irrelevant transitions?
- Problems: complexity, optimality
- Where do FSMs come from?
- PA and planning

State space

State space $\mathcal{S}$

- **set of states, created by n boolean state variables**
- **set of transitions between these states**

Search problem in $\mathcal{S}$:
one initial state, set of accepting states

Solution to search problem:
sequence of transitions that connect the initial to some accepting state

Size of the state space is 2^n
too large to be searched directly

$\rightsquigarrow$ **exploit substructures**

FSMs as substructures of a state space

Let $\mathcal{V}$ be a subset of state variables

If in every state of a state space $\mathcal{S}$, exactly one $v \in \mathcal{V}$ is true $\leftrightarrow$ $(\mathcal{V}, T, v_{\mathcal{I}}, V_{\mathcal{G}})$ finite state machine (FSM) of $\mathcal{S}$, where

- $\mathcal{V}$: set of state variables
- T a bag of transitions, each is element of $\mathcal{V} \times \mathcal{V}$
- $v_{\mathcal{I}} \in \mathcal{V}$, $V_{\mathcal{G}} \subseteq \mathcal{V}$: initial state variable and accepting state variables
- no input, no output

Path in FSM M: sequence of transitions in M

Solution to M: path in M that connects $v_{\mathcal{I}}$ to $v \in V_{\mathcal{G}}$

Solving a collection of FSMs

For simplicity let us assume:
Every state variable is part of an FSM

If FSMs are independent of each other, then

- **Combination of solutions to FSMs is a solution to search problem**
- **For this, any solution to an FSM is acceptable and**
- **Any order between the transitions of different paths is acceptable**

In general, FSMs are synchronized:

- **State variables can be shared among FSMs**
- **Transitions can be shared among FSMs**
- **Transitions can have state variables of other FSMs as precondition**

The side-effects of paths: Sidepaths

Synchronization of FSMs $\rightsquigarrow$ side-effects of paths

- A path has the truth of certain state variables of other FSMs as precondition
- Its execution changes the truth of these dependent FSMs

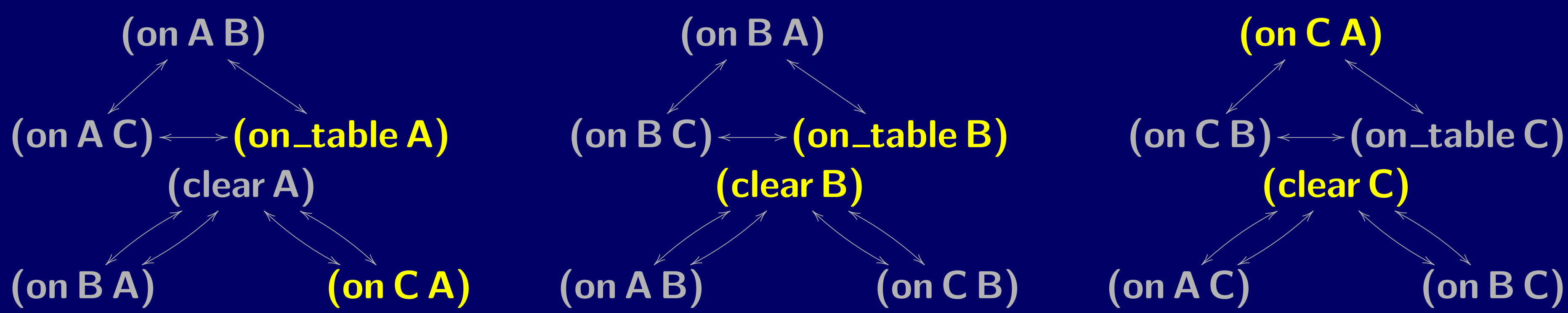
The side-effect of a path on another FSM is similar to
the effect of execution a path in this FSM: Sidepath

Still, solution to search problem is a combination of solutions to FSMs, but:

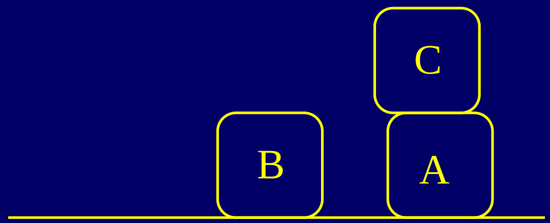
- Only specific solutions to FSMs can be combined
- Order between the transitions of different paths is crucial

Example of dependent FSMs

blocksworld problem with three blocks

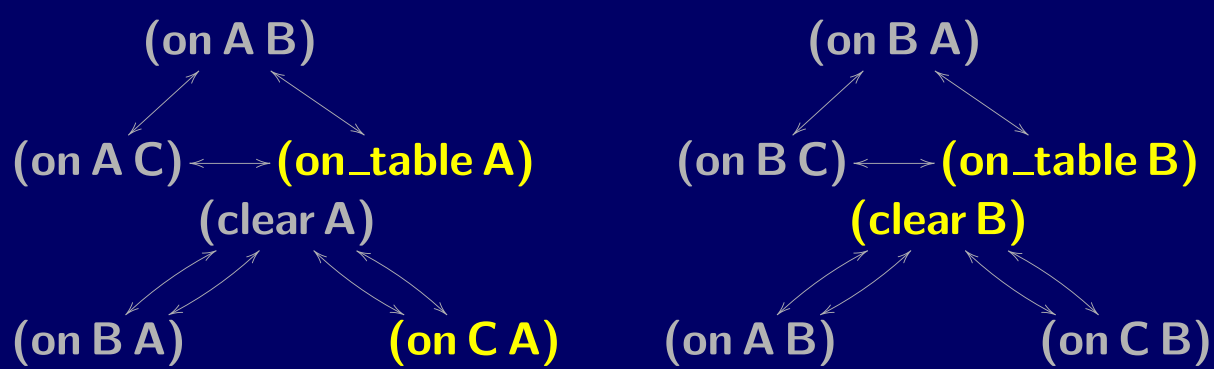


(move_onto_table C A)
(move_from_table B C)
(move_from_table A B)

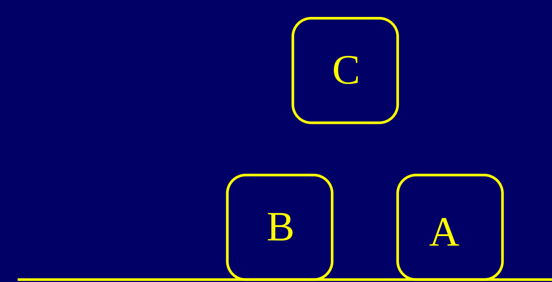


Example of dependent FSMs

blocksworld problem with three blocks



(move_onto_table C A)
(move_from_table B C)
(move_from_table A B)



Example of dependent FSMs

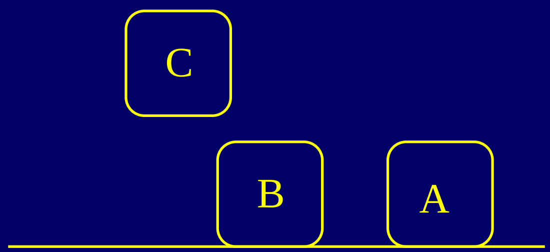
blocksworld problem with three blocks

(on A B)
(on A C) ⟷ (on_table A)
(clear A)
(on B A) (on C A)

(on B A)
(on B C) ⟷ (on_table B)
(clear B)
(on A B) (on C B)

(on C A)
(on C B) ⟷ (on_table C)
(clear C)
(on A C) (on B C)

(move_onto_table C A)
(move_from_table B C)
(move_from_table A B)



Example of dependent FSMs

blocksworld problem with three blocks

(on A B)

(on A C) ⟷ **(on_table A)**

(clear A)

(on B A) (on C A)

(on B A)

(on B C) ⟷ **(on_table B)**

(clear B)

(on A B) (on C B)

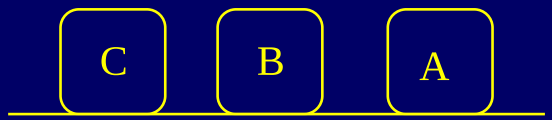
(on C A)

(on C B) ⟷ **(on_table C)**

(clear C)

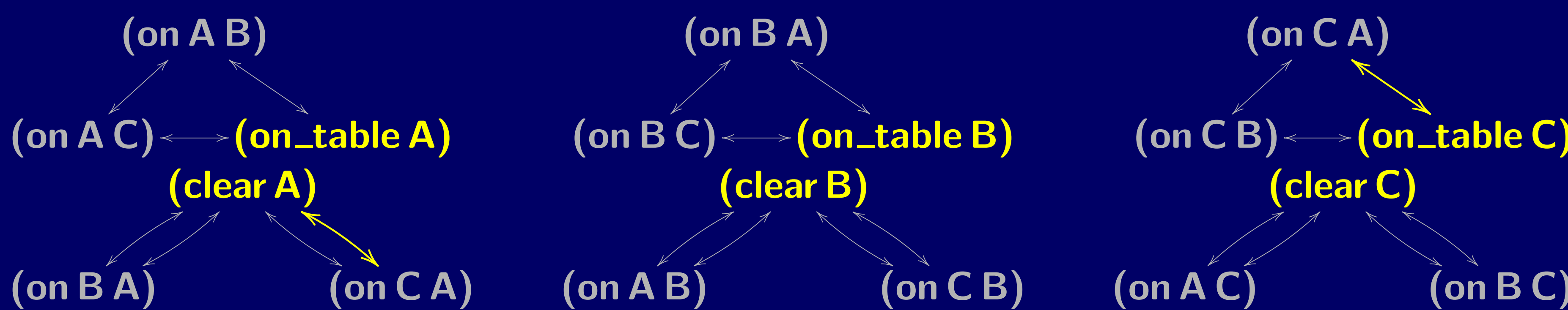
(on A C) (on B C)

(move_onto_table C A)
(move_from_table B C)
(move_from_table A B)

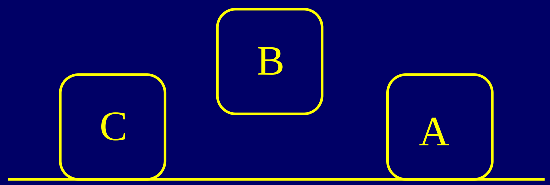


Example of dependent FSMs

blocksworld problem with three blocks

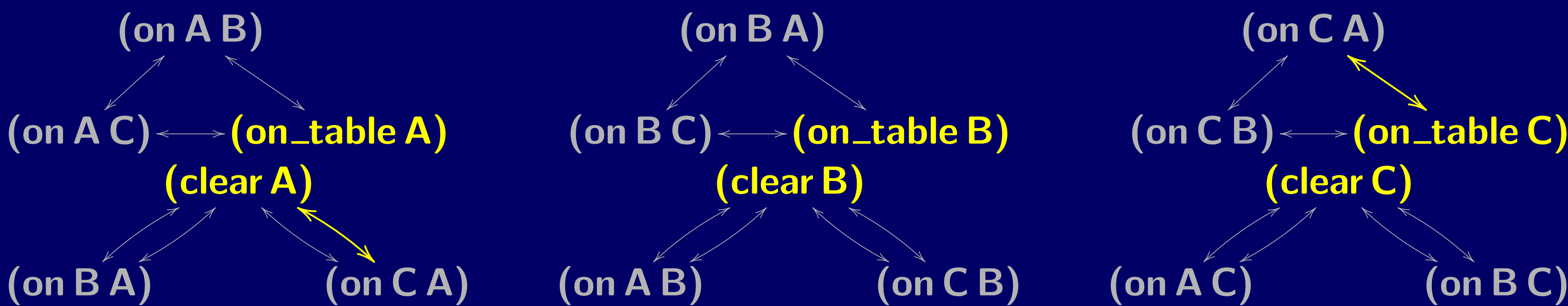


(move_onto_table C A)
(move_from_table B C)
(move_from_table A B)

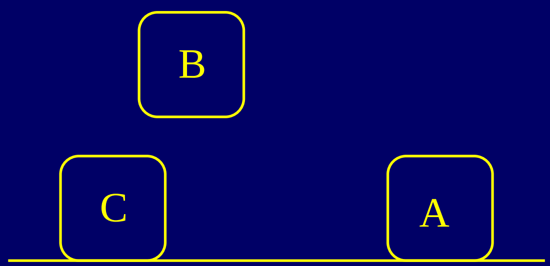


Example of dependent FSMs

blocksworld problem with three blocks

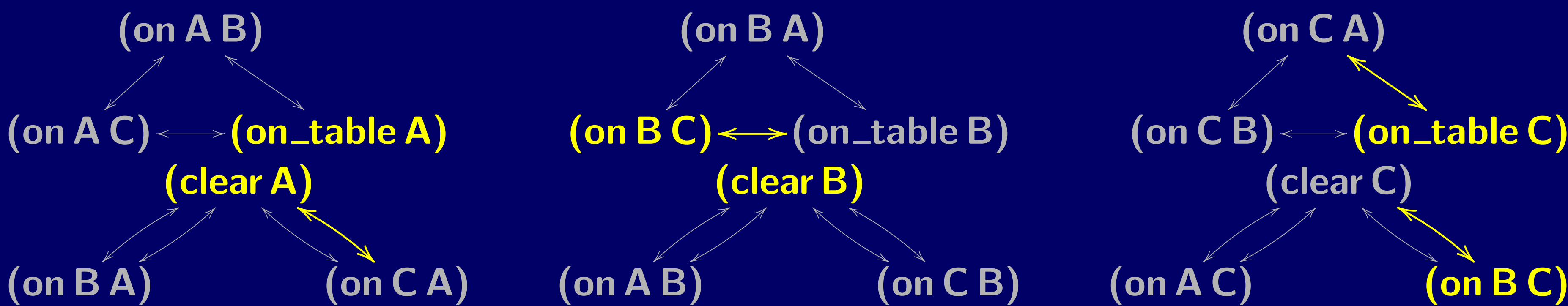


(move_onto_table C A)
(move_from_table B C)
(move_from_table A B)

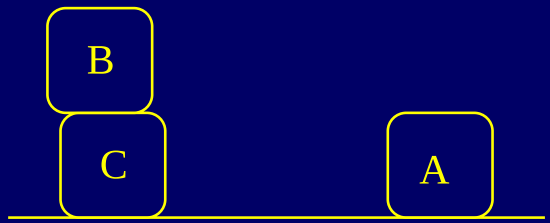


Example of dependent FSMs

blocksworld problem with three blocks

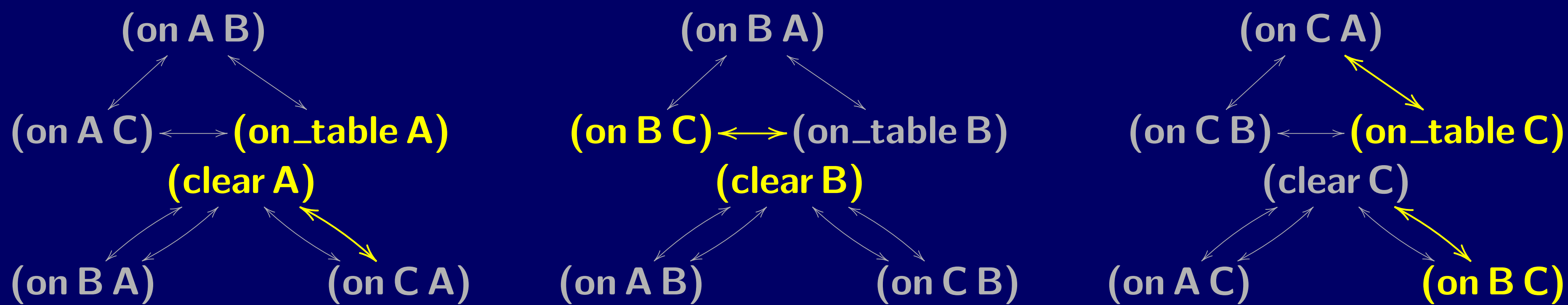


(move_onto_table C A)
(move_from_table B C)
(move_from_table A B)

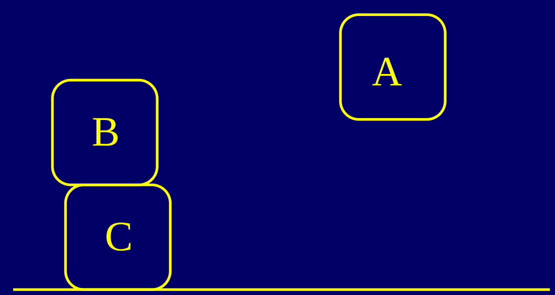


Example of dependent FSMs

blocksworld problem with three blocks

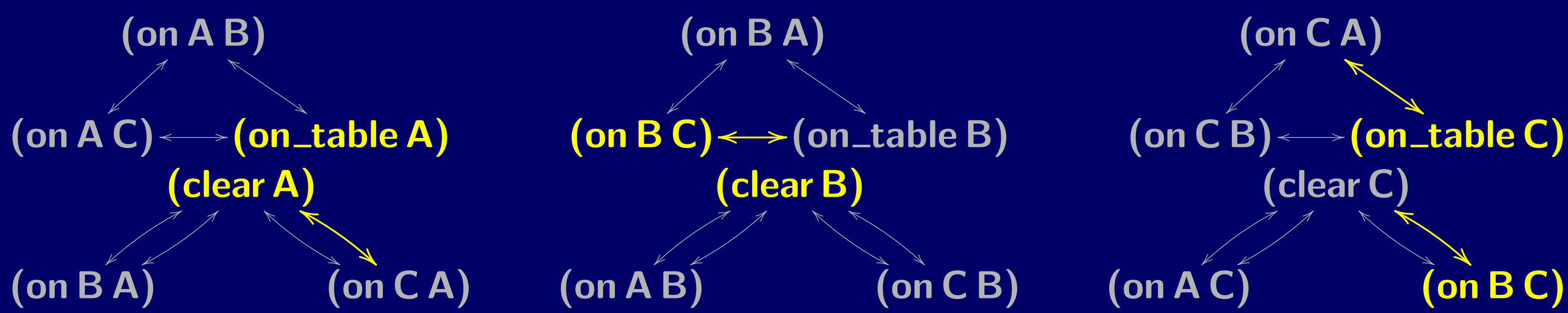


(move_onto_table C A)
(move_from_table B C)
(move_from_table A B)

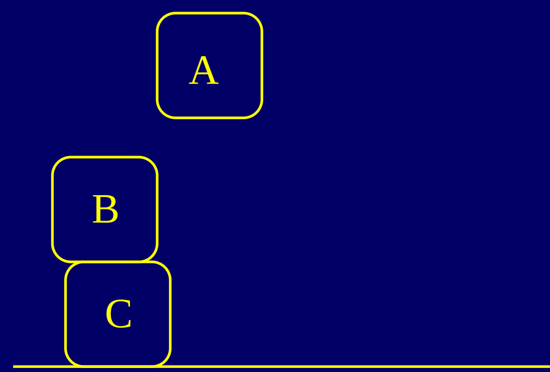


Example of dependent FSMs

blocksworld problem with three blocks

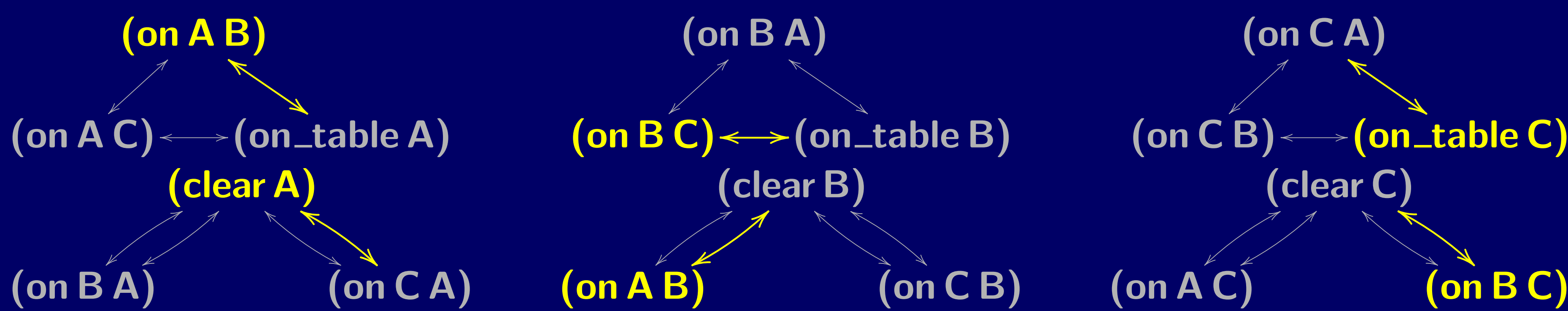


(move_onto_table C A)
(move_from_table B C)
(move_from_table A B)

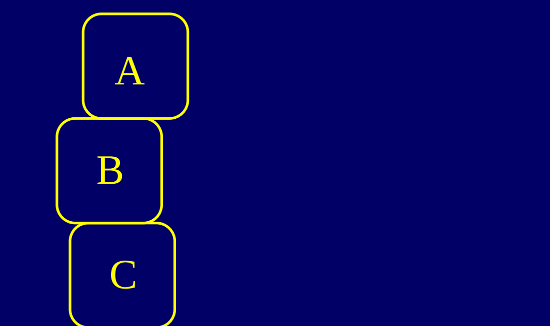


Example of dependent FSMs

blocksworld problem with three blocks



(move_onto_table C A)
(move_from_table B C)
(move_from_table A B)



How to use FSMs for reducing the state transition graph?

Finding a single solution for each FSM st. they can be combined
as hard as
finding a solution to the search problem

better: Identify paths that are possibly relevant for a solution
$\rightsquigarrow$ unused transitions are not relevant

When is a path relevant?

- It can be part of a solution
- There is no "better" path

Replaceability of paths

When is a path better than another path?
If it can be used anywhere the other one can

A path P_1 can replace a path P_2, if

- **They connect the same state variables**
- **For every sidepath of P_1 there is a sidepath of P_2, st. both connect the same state variables**
- **P_1 can be partitioned, st. the i-th transition of P_2 can be exchanged with the i-th partition**

Loosely spoken:

- **P_1 has "the same" side-effects than P_2**
- **They do not "occur earlier"**
- **They do not "end later"**

Which paths might be part of a solution?

A path P in an FSM has the potential to be relevant, if it connects

- $v_{\mathcal{I}}$ to a $v \in V_{\mathcal{G}}$

P has side-effects on other FSMs.
Therefore, a path has the potential to be relevant, if it connects

- $v_{\mathcal{I}}$ to a beginning of a sidepath (of a relevant path)
- The end of a sidepath to a $v \in V_{\mathcal{G}}$
- The end of a sidepath to a beginning of a sidepath

Idea: To find all relevant sidepaths, use a fixpoint computation

How to find a fixpoint of relevant paths

- Maintain sets
  - ⋆ $\mathcal{B}$ and $\mathcal{E}$ of state variables
  - ⋆ $\mathcal{P}$ of paths

  Initially, $\mathcal{B}$ ($\mathcal{E}$) is the initial (the goal) state variable, $\mathcal{P} = \emptyset$
- Construct paths, which connect state variables of $\mathcal{B}$ with those of $\mathcal{E}$
- Add paths to $\mathcal{P}$, which are not replaceable
- Add new beginnings of sidepaths to $\mathcal{E}$ and new endings to $\mathcal{B}$
- Terminate, if all pairs in $\mathcal{B} \times \mathcal{E}$ are considered

The transitions of paths in $\mathcal{P}$ are sufficient to find a solution to the search problem

$\rightsquigarrow$ all other transitions are irrelevant

Problems and (possible) solutions

- Reduction does not preserve optimality
- Number of paths in an FSM can be high (or even infinite)
- Number of FSMs can be exponential in the number of state variables
- Reduction depends on search problem, not only on search space

Problem: Optimality

Longer paths can replace shorter ones but not vice versa
$\rightsquigarrow$ Reduction is solution preserving, but does not preserve optimality

Solution: Accept shorter paths although they can be replaced

Problem: Large number of paths

Paths can be cyclic
The number of non-cyclic paths can be factorial in the size of an FSM

but: For the fixpoint computation, there is no need to consider

- Cyclic paths
- Paths that traverse a state variable of $\mathcal{B}$ and then one of $\mathcal{E}$

In addition, we can

- reject paths $P \circ P'$ and $P' \circ P$, if P got rejected

If the length of the longest relevant path is constant in problem size, then the number of relevant paths is polynomial

$\rightsquigarrow$ enumerate short paths first, reject unnecessary paths

Problem: Large number of FSMs

- The number of FSMs can be exponential in the number of state variables

  More precisely:

  ⋆ State variables can be partitioned,
  st. FSMs are unions of some partitions
  ⋆ number of FSMs is exponential in the number of partitions
  ⋆ Only some combinations of partitions are FSMs
  state space has additional structure which can be exploited

  $\rightsquigarrow$ PA on partitions instead of FSMs

- There can be many similar FSMs in a state space

  $\rightsquigarrow$ Use of parameterized FSMs and paths

  E. g. a blocksworld with n blocks has $2n$ FSMs,
  but only 2 parameterized FSMs

Problem: Reduction is problem-dependent

(possible) solution:

- **Use of parameterized FSMs**
- **Calculate parameterized fixpoint from instance**
- **Instantiate fixpoint with new problems**

Where do FSMs come from?

- **Why do there exists FSMs as substructure of the state space?**

  Things can be only at one place
  Objects have exactly one property out of a set of possible ones

- **How do we find FSMs?**

  In planning, FSMs are state invariants
  They can be found by preprocessing, e. g. TIM and others

Path Analysis and Planning

In planning

- state variables are facts
- transitions are actions

Problem: Planning problems can violate assumptions:

- Not all facts are part of an FSM
- Not all actions are a transition of an FSM
- Transitions can lead to a state of the search problem, where all state variables are false

Path analysis can adopt to these special cases

Work on path analysis is motivated by planning

- First implementation shows that the idea works
- Only parameterized implementation has potential to be fast enough