Probabilistic Bisimulation

Seminar *Probabilistic Models of Concurrency*

July 2020

Johannes Raufeisen

Software Modeling and Verification Chair
RWTH AACHEN UNIVERSITY

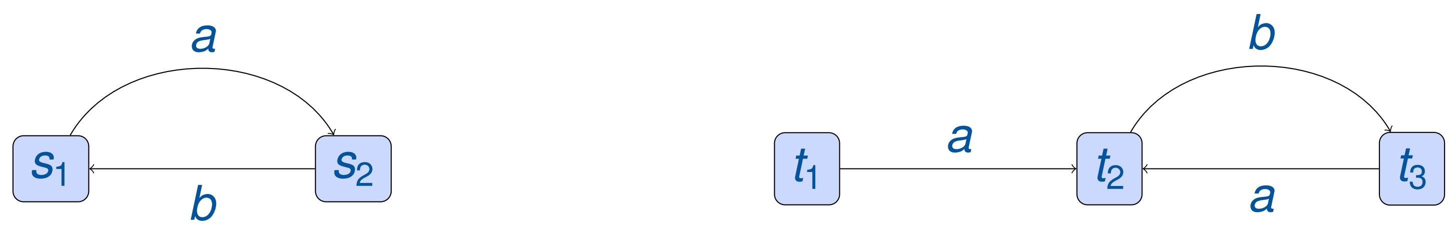
Outline

Outline

Bisimulation

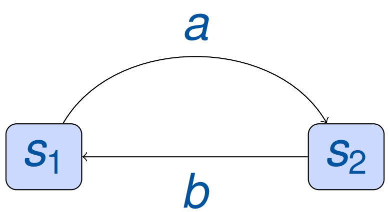
Motivation

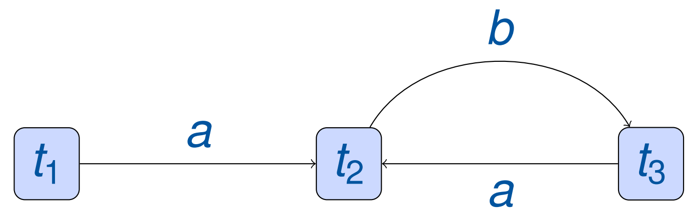
- Determine equivalence of transition systems (TS) under a certain logic
- Reducing size of TS while maintaining logical properties
- Optimize model checking performance



Two bisimilar transition systems

Introduction





Definition - Bisimulation for an ordinary TS

Let $TS = (S, Act, \rightarrow, I, AP, L)$ be a transition system. A bisimulation for TS is a binary relation $\mathcal{R}$ such that for all $(s_1, s_2) \in \mathcal{R}$:

- $L(s_1) = L(s_2)$
- If $s_1' \in \text{Post}(s_1)$, then there exists an $s_2' \in \text{Post}(s_2)$ with $(s_1', s_2') \in \mathcal{R}$
- If $s_2' \in \text{Post}(s_2)$, then there exists an $s_1' \in \text{Post}(s_1)$ with $(s_1', s_2') \in \mathcal{R}$

States s_1 and s_2 are bisimulation-equivalent (or bisimilar), denoted $s_1 \sim_{TS} s_2$, if there exists a bisimulation $\mathcal{R}$ for TS with $(s_1, s_2) \in \mathcal{R}$.

Introduction

Aim of this presentation

1. Introduce a variant of probabilistic automata (PLTS)
2. Define probabilistic bisimulation for these automata
3. Present an algorithm (Groote, Veduzsco, de Vink) to efficiently compute probabilistic bisimulation

Software Modeling and Verification Chair
RWTH AACHEN UNIVERSITY

Introduction

Aim of this presentation

1. Introduce a variant of probabilistic automata (PLTS)
2. Define probabilistic bisimulation for these automata
3. Present an algorithm (Groote, Veduzsco, de Vink) to efficiently compute probabilistic bisimulation

Why is this relevant?

Probabilistic bisimulation is constructed to maintain equivalence under the logic PCTL (covered in another presentation)
$\longrightarrow$ Use the quotient TS to reduce the amount of states
$\longrightarrow$ Useful for Model Checking

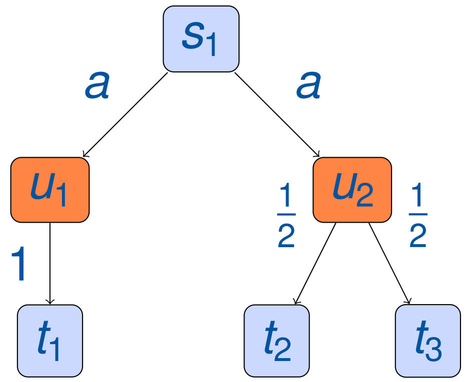
Outline

Introduction

Probabilistically labeled transition system (PLTS)

A probabilistically labeled transition system (PLTS) with a set of actions *Act* is a tuple $\mathcal{A} = (S, \rightarrow)$ such that

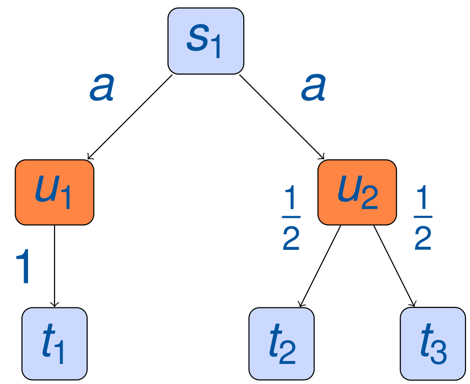
- S is a finite set, containing the states of the system
- $\mathcal{D}(S)$ is the set of probabilistic distributions over S
- $\rightarrow \subseteq S \times Act \times \mathcal{D}(S)$ is a finite relation, containing the transitions of the system



Introduction

Example

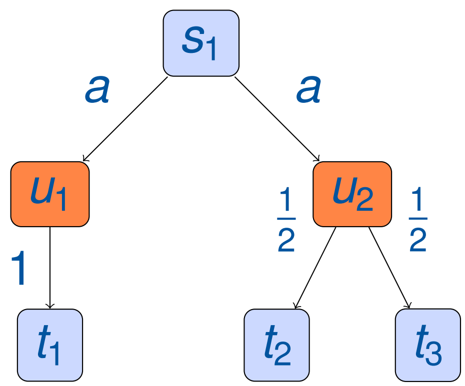
- $S = \{s_1, t_1, t_2, t_3\}$
- $Act = \{a\}$
- $\rightarrow = \{(s_1, a, u_1), (s_1, a, u_2)\}$
- u_1 denotes the distribution $u_1 : S \rightarrow [0, 1]$ with $u_1(t_1) = 1$
- u_2 denotes the distribution $u_2 : S \rightarrow [0, 1]$ with $u_2(t_2) = 0.5$ and $u_2(t_3) = 0.5$



Introduction

Notation

- S is called the set of action states
- For $f \in \mathcal{D}(S)$, we call u_f the associated probabilistic state
- The set of prob. states is called U
- We define $f[T] = \sum_{s \in T} f(s)$ for subsets $T \subseteq S$



Example of a PLTS

Definition - Probabilistic bisimulation

Let $\mathcal{A} = (S, \rightarrow)$ be a PLTS.

An equivalence relation $\sim$ on S is called a **probabilistic bisimulation** if and only if:

- For all states $s, t \in S$ such that $s \sim t$ and $s \xrightarrow{a} f$ for some $a \in \mathrm{Act}$ and $f \in \mathcal{D}(S)$ there is a $g \in \mathcal{D}(S)$ such that $t \xrightarrow{a} g$ and $f[B] = g[B] \quad \forall B \in S/\!\sim$

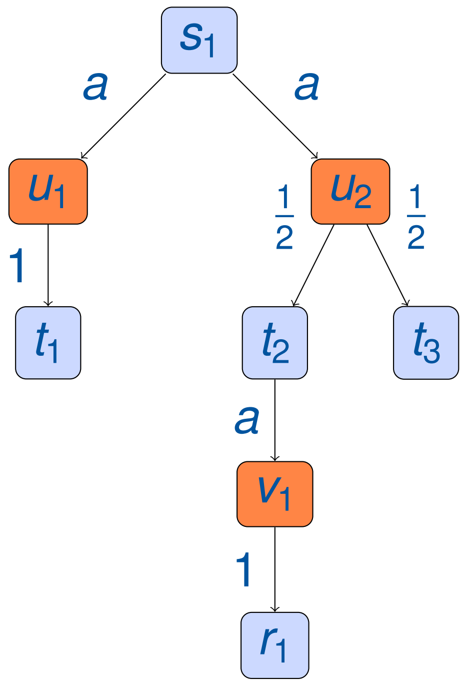
Two **action states** $s, t \in S$ are called probabilistically bisimilar if and only if:

- There is a probabilistic bisimulation $\sim$ such that $s \sim t$

Two **probabilistic states** u_f, u_g are called probabilistically bisimilar if and only if:

- There is a probabilistic bisimulation $\sim$ such that for all sets $B \in S/\!\sim$ it holds that $f[B] = g[B]$

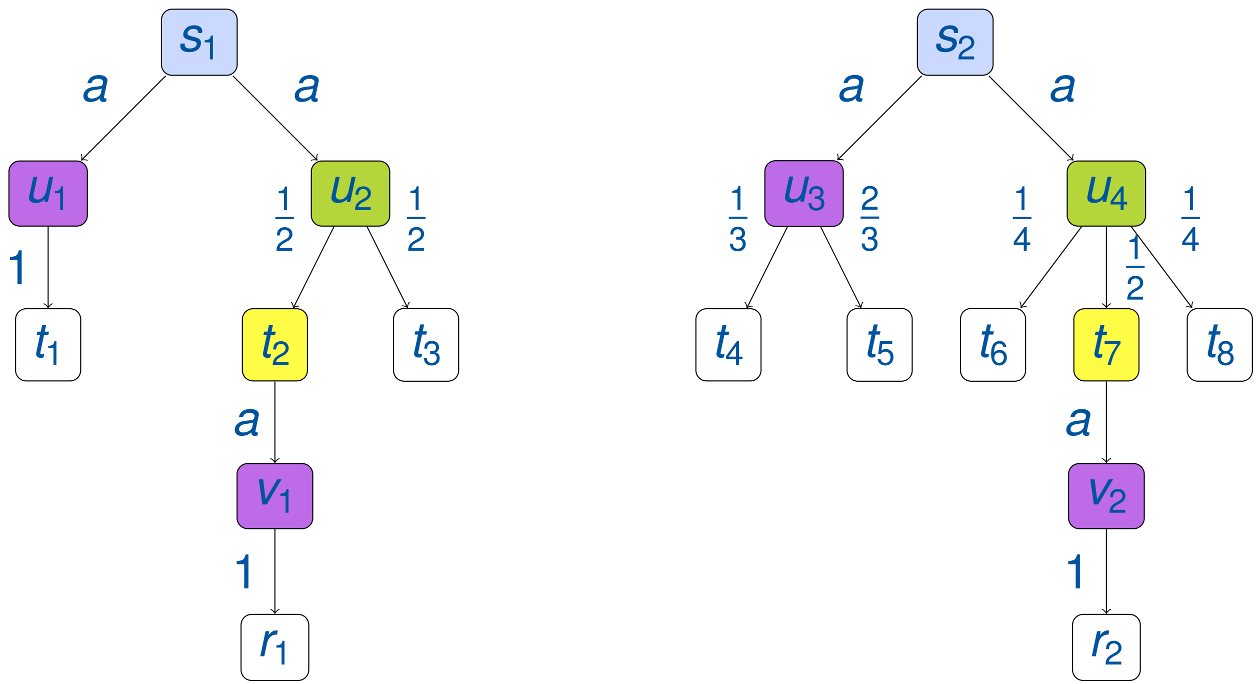
Probabilistic bisimulation



Two probabilistically bisimilar nondeterministic transition systems

Introduction

Probabilistic bisimulation



Two probabilistically bisimilar nondeterministic transition systems

Outline

Outline

Algorithm

General remarks

- View bisimilarity as a partition on the set $S \cup U$
- Partition refinement algorithm
- Keep two partitions:
  - $\mathcal{C}$ - The set of constellations
  - $\mathcal{B}$ - The set of blocks which will later be our bisimulation
  - $\mathcal{B}$ will always be a refinement of $\mathcal{C}$

Algorithm

General remarks

- View bisimilarity as a partition on the set $S \cup U$
- Partition refinement algorithm
- Keep two partitions:
  - $\mathcal{C}$ - The set of constellations
  - $\mathcal{B}$ - The set of blocks which will later be our bisimulation
  - $\mathcal{B}$ will always be a refinement of $\mathcal{C}$

Structure of the algorithm

1. Initialize $\mathcal{B}$ and $\mathcal{C}$ with trivial partitions
2. While $\mathcal{B}$ is not equal to $\mathcal{C}$
   - Refine the partitions $\mathcal{B}$ and $\mathcal{C}$ according to certain criteria
   - Both partitions $\mathcal{B}$ and $\mathcal{C}$ will be more fine-grained
   - $\mathcal{B}$ will always be a refinement of $\mathcal{C}$

Algorithm

Preliminaries

- A set of action states B is called **stable** under a set of probabilistic states $C \subseteq U$ iff for all actions $a \in Act$ it holds that:

$$s \xrightarrow{a} C \iff t \xrightarrow{a} C \quad \forall s, t \in B$$

- A set of probabilistic states $B \subseteq U$ is called **stable** under a set of action states $C \subseteq S$ iff

$$u[C] = v[C] \quad \forall u, v \in B$$

- A partition $\mathcal{B}$ is called stable under another partition $\mathcal{C}$, if each block $B \in \mathcal{B}$ is stable under all sets of action states or probabilistic states, respectively.

Algorithm

Algorithm 1 Abstract partition refinement algorithm for probabilistic bisimulation

function PARTITION REFINEMENT
- $\mathcal{C} := \{S, U\}$
- $\mathcal{B} := \{U\} \cup \{S_A \mid A \subseteq Act\}$
- where $S_A = \left\{ s \in S \mid \forall a \in Act \left(\exists u \in U : s \xrightarrow{a} u \iff a \in A \right) \right\}$

Initialization

- **while** $\mathcal{C}$ contains a non-trivial constellation C **do**

Termination condition

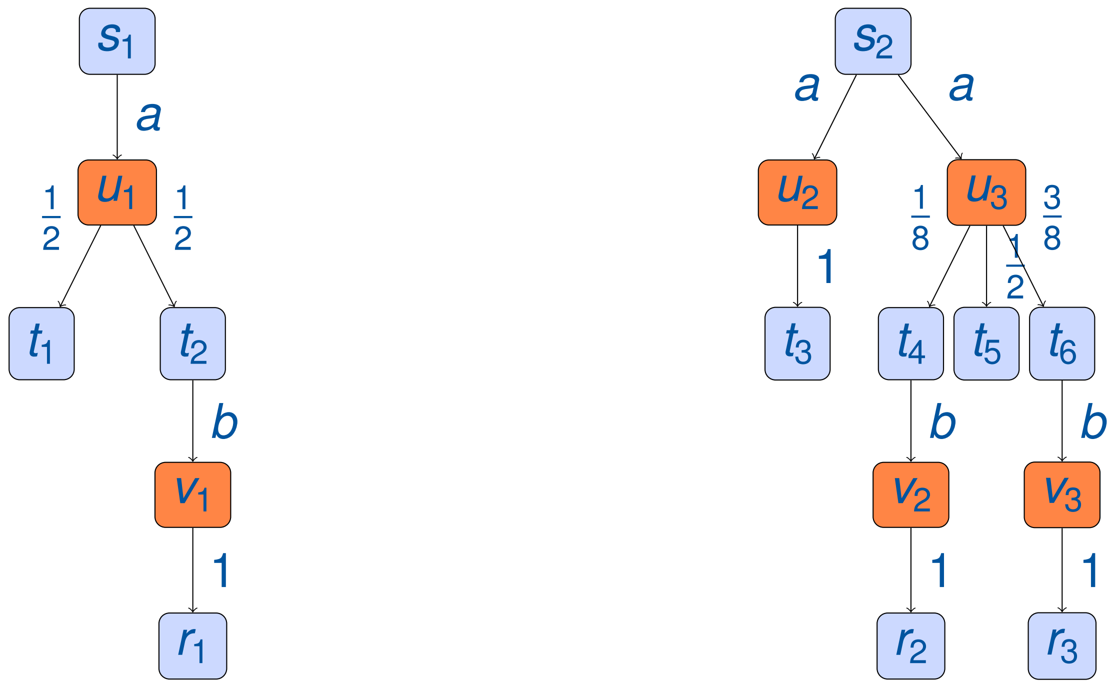
- choose block B_C from $\mathcal{B}$ in C
- replace in $\mathcal{C}$ constellation C by B_C and $C \backslash B_C$
- **if** C contains probabilistic states **then**
  - **for all** blocks B of action states in $\mathcal{B}$ unstable under B_C or $C \backslash B_C$ **do**
    - refine $\mathcal{B}$ by splitting B
    - into blocks of states with the same actions into B_C and $C \backslash B_C$
- **else**
  - **for all** blocks B of probabilistic states in $\mathcal{B}$ unstable under B_C **do**
    - refine $\mathcal{B}$ by splitting B
    - into blocks of states with equal probabilities into B_C

Refinement by splitting unstable sets

- **return** $\mathcal{B}$

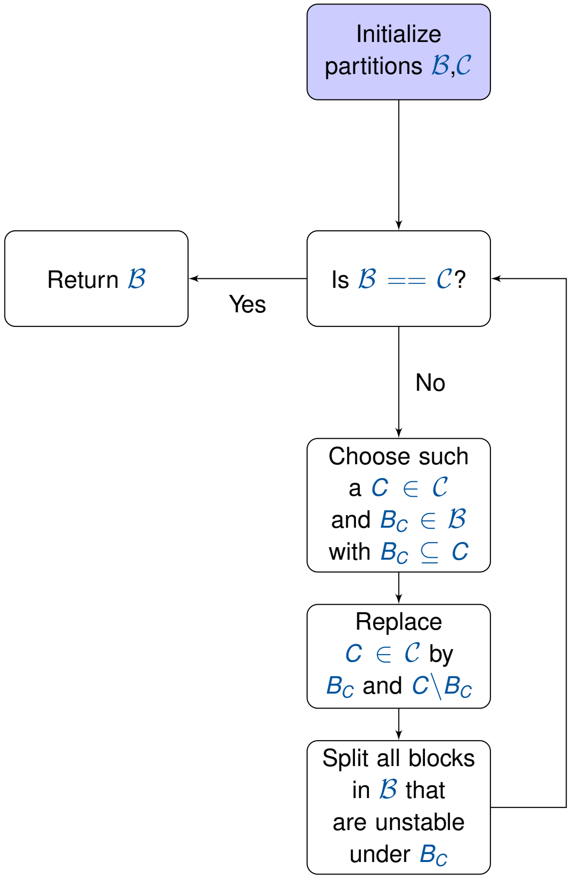
Outline

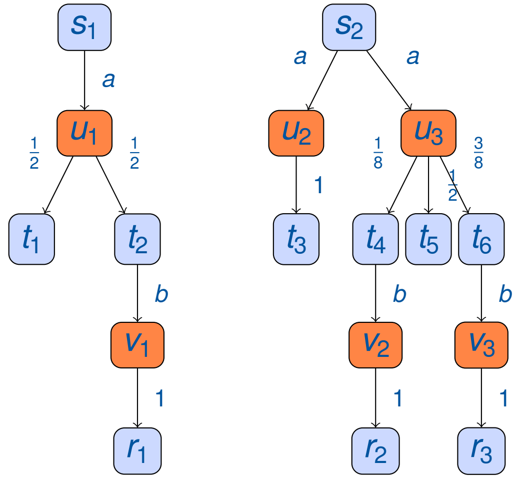
Example computation



Example PLTS for algorithm 1

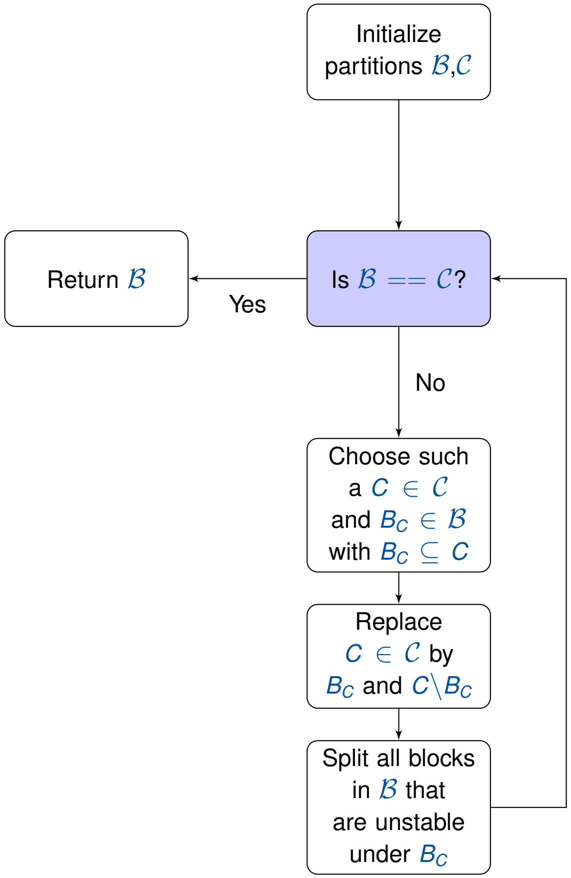
Algorithm

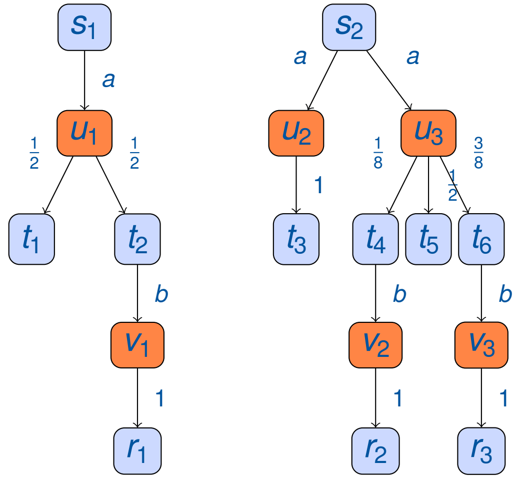




$$\mathcal{C} = \{U, S\}$$
$$= \{\{u_{1-3}, v_{1-3}\}, \{s_1, s_2, t_{1-6}, r_{1-3}\}\}$$
$$\mathcal{B} = \{U, S_{\{a\}}, S_{\{b\}}, S_{\varnothing}\}$$
$$= \{\{u_{1-3}, v_{1-3}\}, \{s_1, s_2\}, \{t_2, t_4, t_6\}, \{t_1, t_3, t_5, r_{1-3}\}\}$$

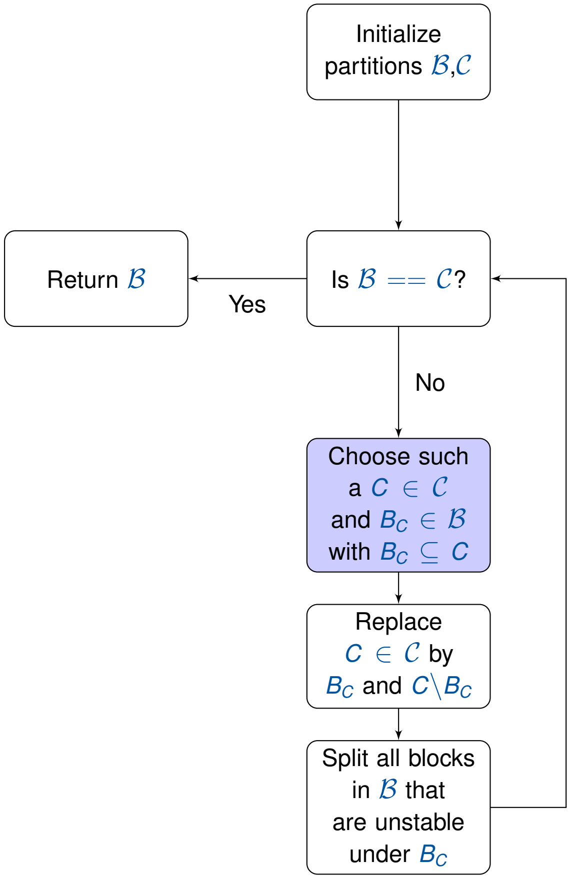
Algorithm

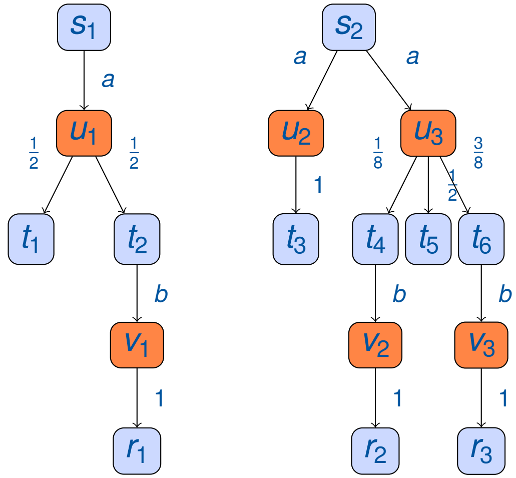




$$\begin{aligned}\mathcal{C} &= \{U, S\} \\ &= \{\{u_{1-3}, v_{1-3}\}, \{s_1, s_2, t_{1-6}, r_{1-3}\}\} \\ \mathcal{B} &= \{U, S_{\{a\}}, S_{\{b\}}, S_\varnothing\} \\ &= \{\{u_{1-3}, v_{1-3}\}, \{s_1, s_2\}, \{t_2, t_4, t_6\}, \{t_1, t_3, t_5, r_{1-3}\}\}\end{aligned}$$

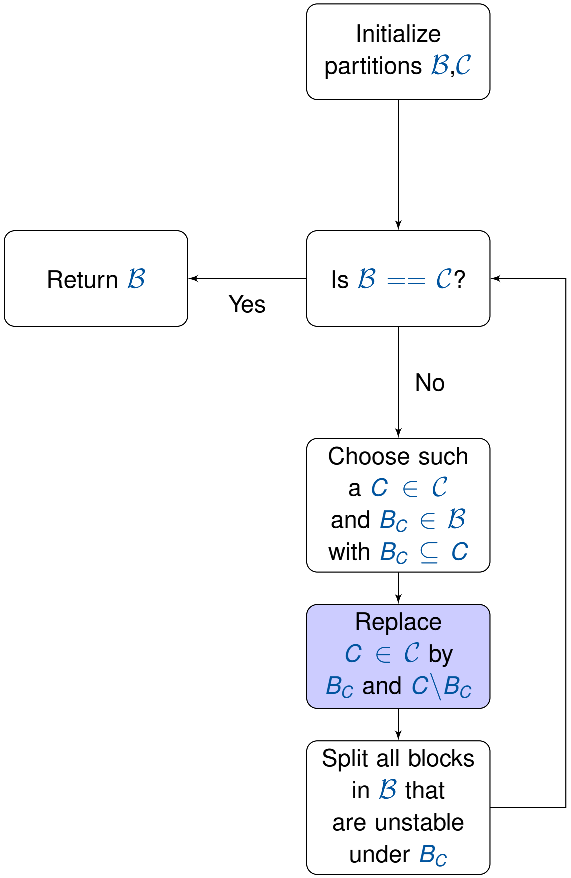
Algorithm

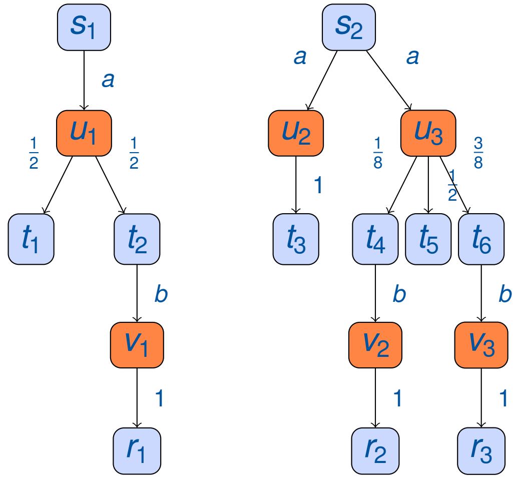




$$
\begin{aligned}
C &= S \\
B_C &= S_{\{a\}} \\
\mathcal{C} &= \{U, S\} \\
&= \{\{u_{1-3}, v_{1-3}\}, \{s_1, s_2, t_{1-6}, r_{1-3}\}\} \\
\mathcal{B} &= \{U, S_{\{a\}}, S_{\{b\}}, S_{\varnothing}\} \\
&= \{\{u_{1-3}, v_{1-3}\}, \{s_1, s_2\}, \{t_2, t_4, t_6\}, \{t_1, t_3, t_5, r_{1-3}\}\}
\end{aligned}
$$

Algorithm





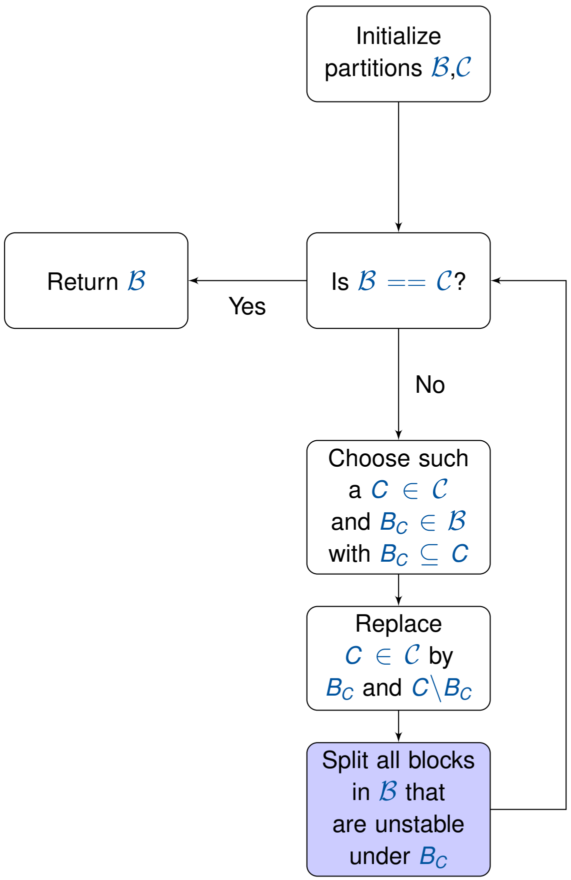
$$C = S$$

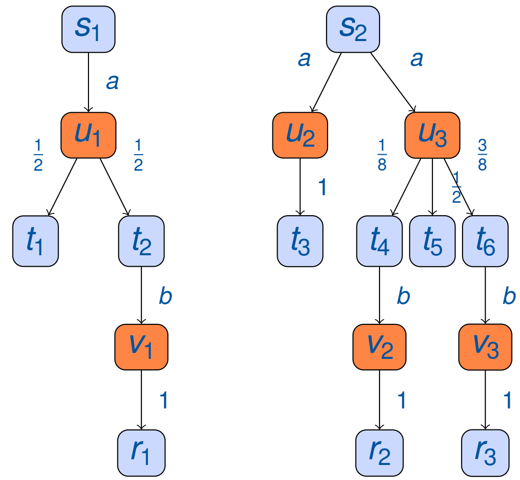
$$B_C = S_{\{a\}}$$

$$\begin{aligned}\mathcal{C} &= \left\{U, S_{\{a\}}, S \setminus S_{\{a\}}\right\} \\ &= \left\{U, \{s_1, s_2\}, \{t_{1-6}, r_{1-3}\}\right\}\end{aligned}$$

$$\begin{aligned}\mathcal{B} &= \left\{U, S_{\{a\}}, S_{\{b\}}, S_{\varnothing}\right\} \\ &= \left\{\{u_{1-3}, v_{1-3}\}, \{s_1, s_2\}, \{t_2, t_4, t_6\}, \{t_1, t_3, t_5, r_{1-3}\}\right\}\end{aligned}$$

Algorithm





$$C = S$$

$$B_C = S_{\{a\}}$$

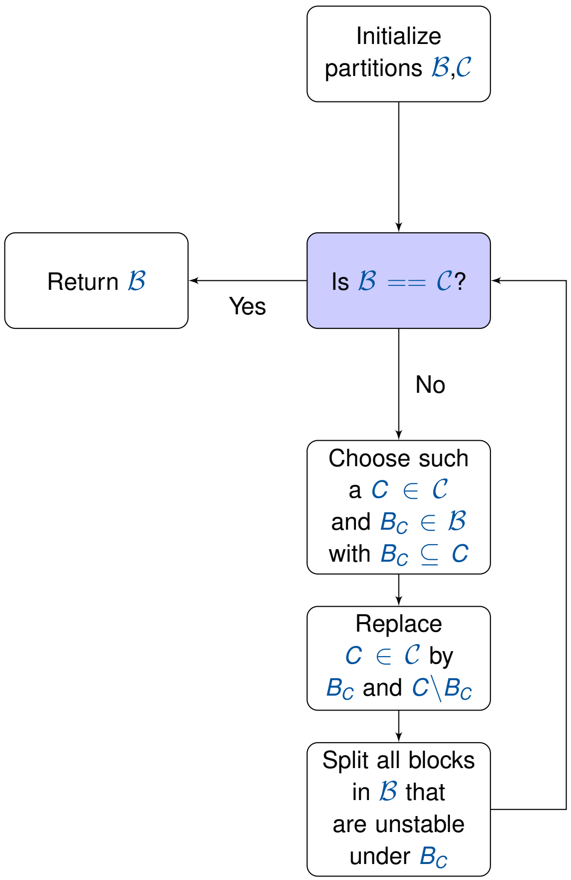
$$\mathcal{C} = \left\{U, S_{\{a\}}, S \setminus S_{\{a\}}\right\}$$

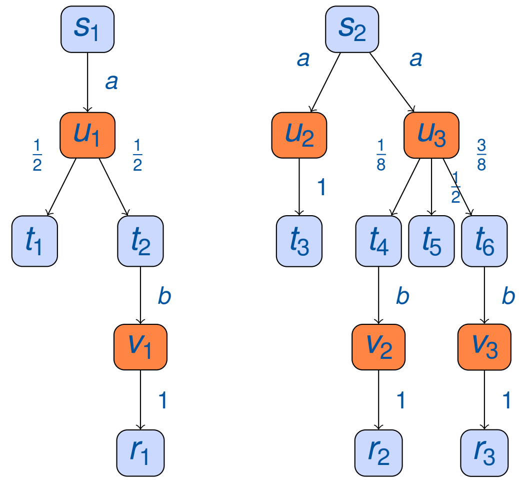
$$= \left\{U, \{s_1, s_2\}, \{t_{1-6}, r_{1-3}\}\right\}$$

$$\mathcal{B} = \left\{U, S_{\{a\}}, S_{\{b\}}, S_{\varnothing}\right\}$$

$$= \left\{\{u_{1-3}, v_{1-3}\}, \{s_1, s_2\}, \{t_2, t_4, t_6\}, \{t_1, t_3, t_5, r_{1-3}\}\right\}$$

Algorithm

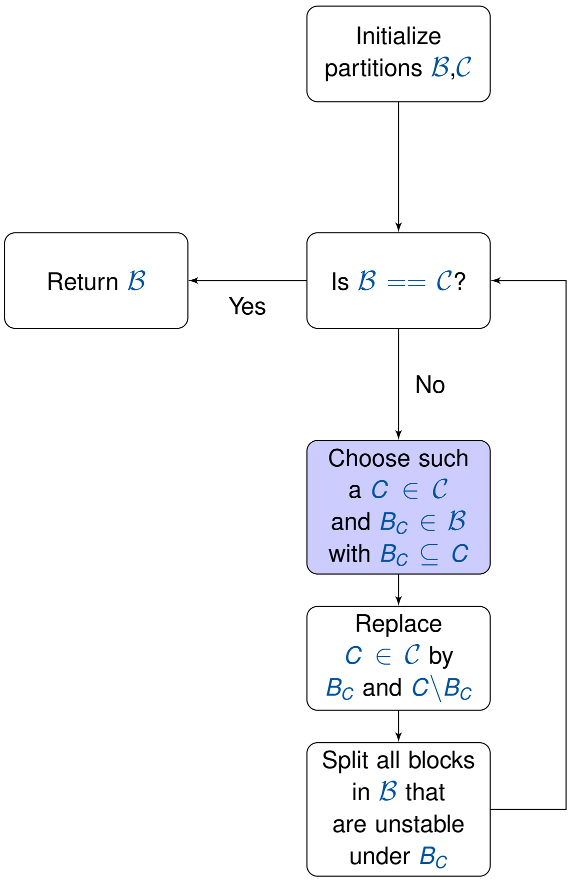


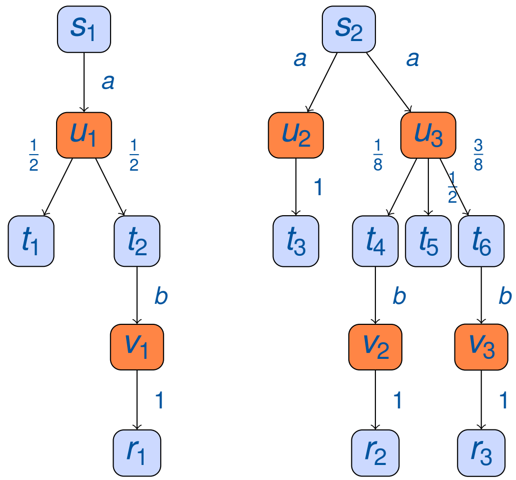


$$\mathcal{C} = \{U, \{s_1, s_2\}, \{t_{1-6}, r_{1-3}\}\}$$

$$\mathcal{B} = \{\{u_{1-3}, v_{1-3}\}, \{s_1, s_2\}, \{t_2, t_4, t_6\}, \{t_1, t_3, t_5, r_{1-3}\}\}$$

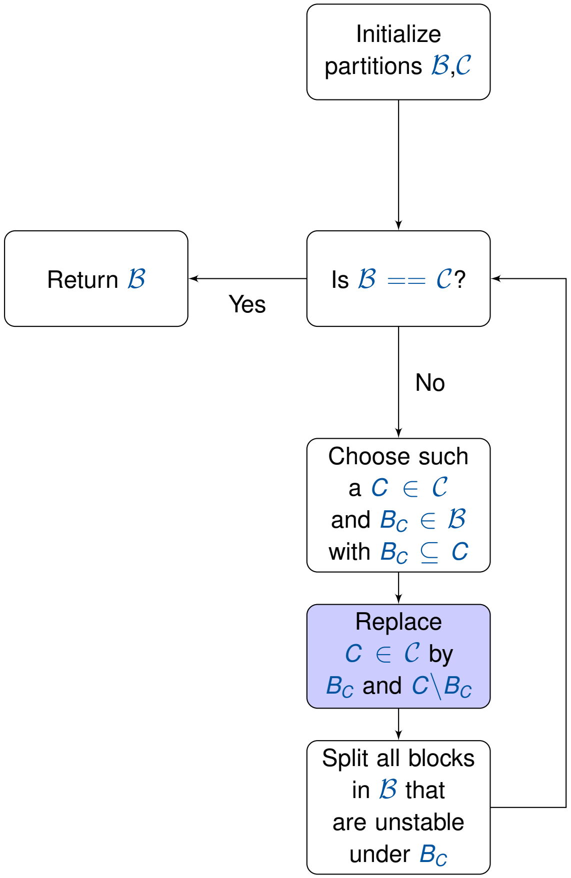
Algorithm

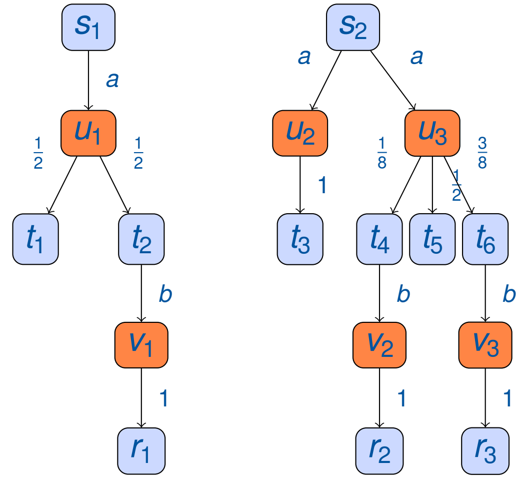




$$C = \{t_{1-6}, r_{1-3}\}$$
$$B_C = \{t_2, t_4, t_6\}$$
$$\mathcal{C} = \{U, \{s_1, s_2\}, \{t_{1-6}, r_{1-3}\}\}$$
$$\mathcal{B} = \{\{u_{1-3}, v_{1-3}\}, \{s_1, s_2\}, \{t_2, t_4, t_6\}, \{t_1, t_3, t_5, r_{1-3}\}\}$$

Algorithm





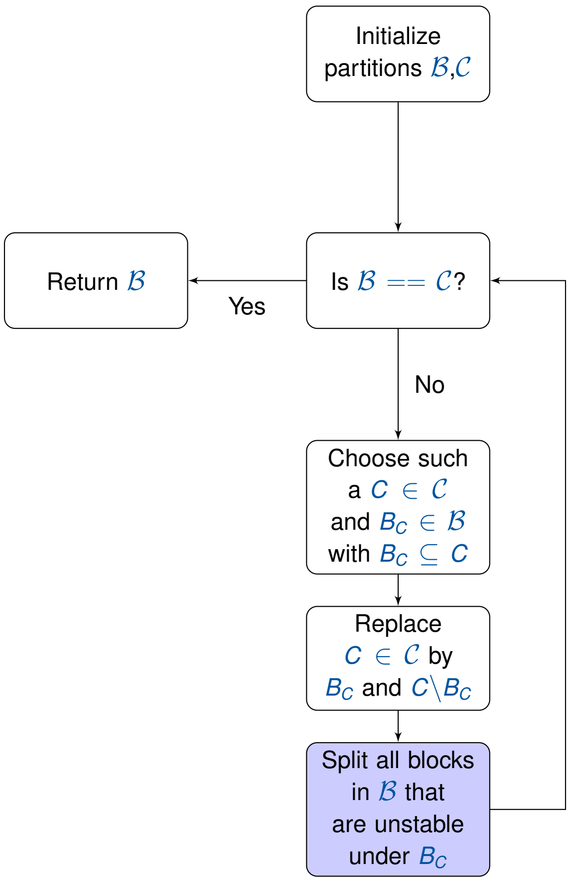
$$C = \{t_{1-6}, r_{1-3}\}$$

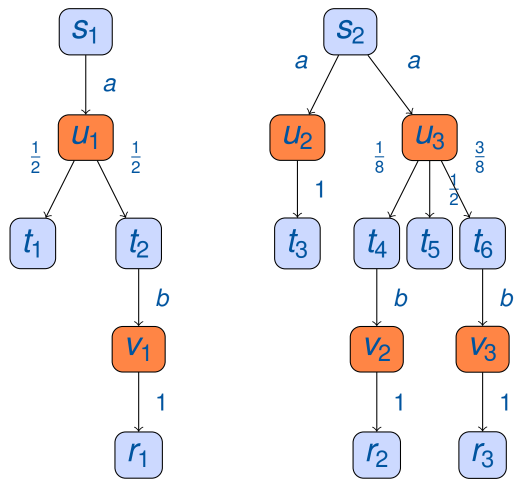
$$B_C = \{t_2, t_4, t_6\}$$

$$\mathcal{C} = \{U, \{s_1, s_2\}, \{t_2, t_4, t_6\}, \{t_1, t_3, t_5, r_{1-3}\}\}$$

$$\mathcal{B} = \{\{u_{1-3}, v_{1-3}\}, \{s_1, s_2\}, \{t_2, t_4, t_5\}, \{t_1, t_3, t_5, r_{1-3}\}\}$$

Algorithm





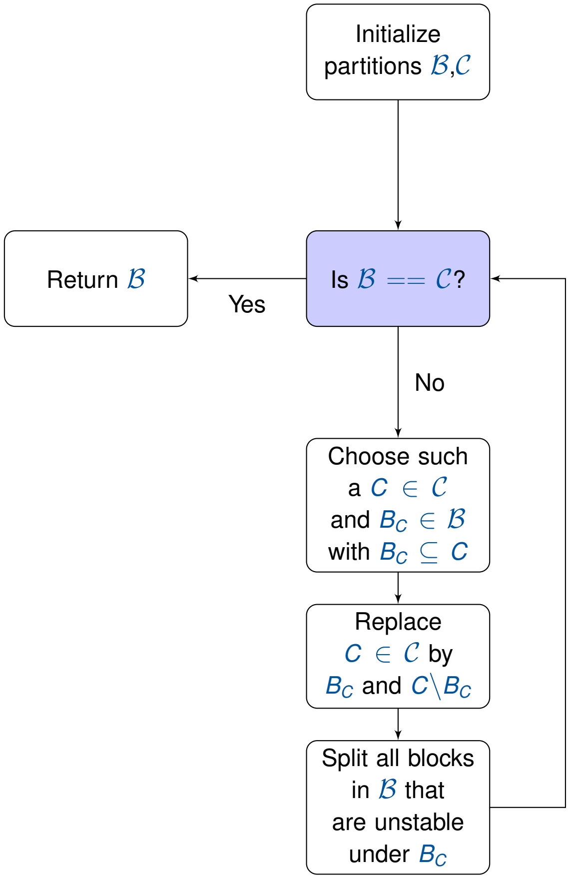
$$C = \{t_{1-6}, r_{1-3}\}$$

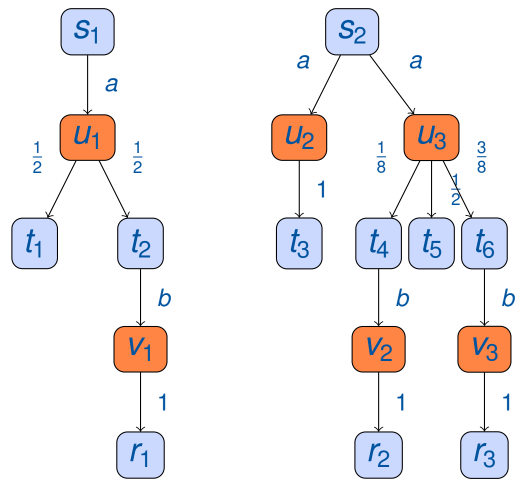
$$B_C = \{t_2, t_4, t_6\}$$

$$\mathcal{C} = \{U, \{s_1, s_2\}, \{t_2, t_4, t_6\}, \{t_1, t_3, t_5, r_{1-3}\}\}$$

$$\mathcal{B} = \{\{u_1, u_3\}, \{u_2, v_{1-3}\}, \{s_1, s_2\}, \{t_2, t_4, t_5\}, \{t_1, t_3, t_6, r_{1-3}\}\}$$

Algorithm

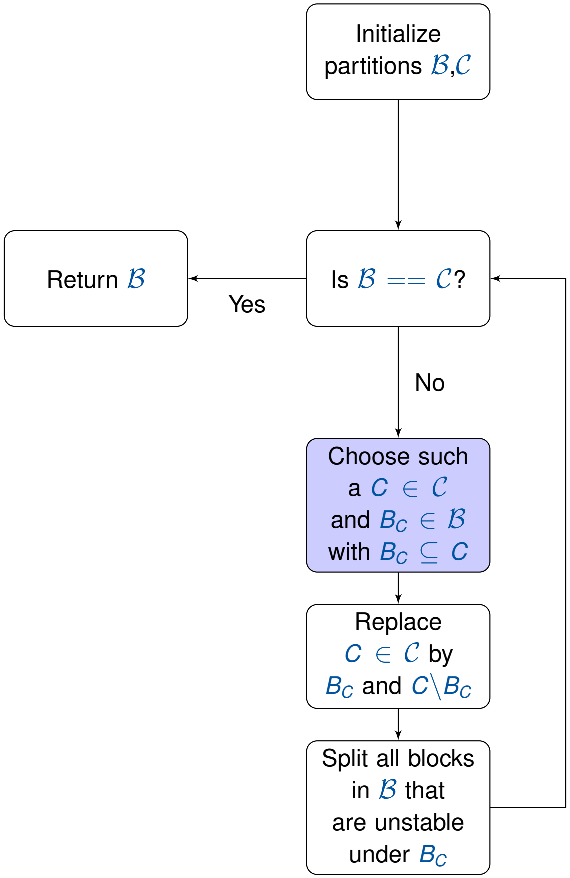


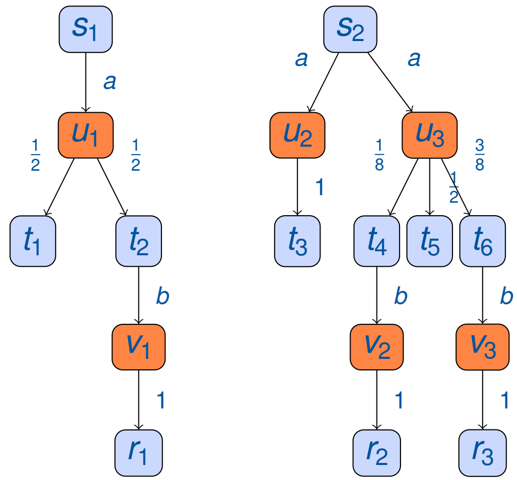


$$\mathcal{C} = \{U, \{s_1, s_2\}, \{t_2, t_4, t_6\}, \{t_1, t_3, t_5, r_{1-3}\}\}$$

$$\mathcal{B} = \{\{u_1, u_3\}, \{u_2, v_{1-3}\}, \{s_1, s_2\}, \{t_2, t_4, t_6\}, \{t_1, t_3, t_5, r_{1-3}\}\}$$

Algorithm





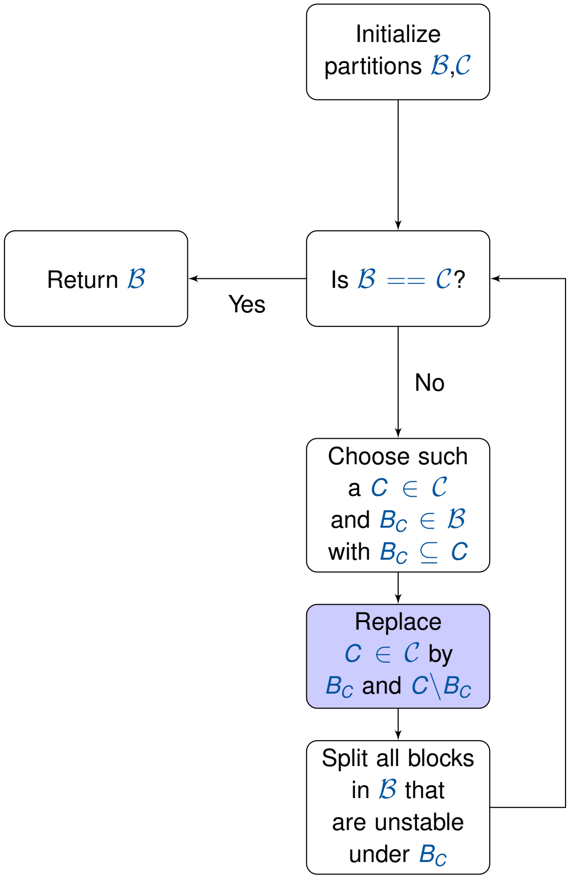
$$C = U$$

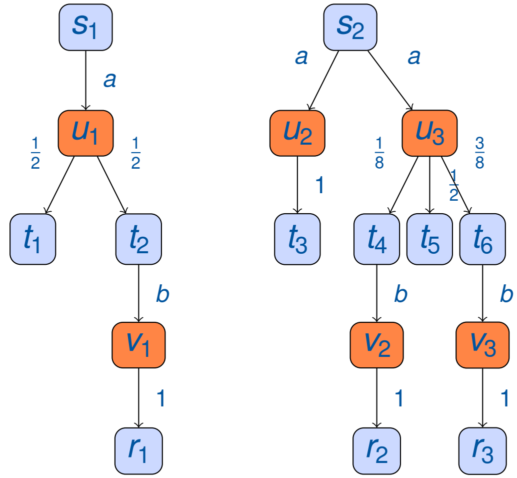
$$B_C = \{u_2, v_{1-3}\}$$

$$\mathcal{C} = \{U, \{s_1, s_2\}, \{t_2, t_4, t_6\}, \{t_1, t_3, t_6, r_{1-3}\}\}$$

$$\mathcal{B} = \{\{u_1, u_3\}, \{u_2, v_{1-3}\}, \{s_1, s_2\}, \{t_2, t_4, t_6\}, \{t_1, t_3, t_5, r_{1-3}\}\}$$

Algorithm





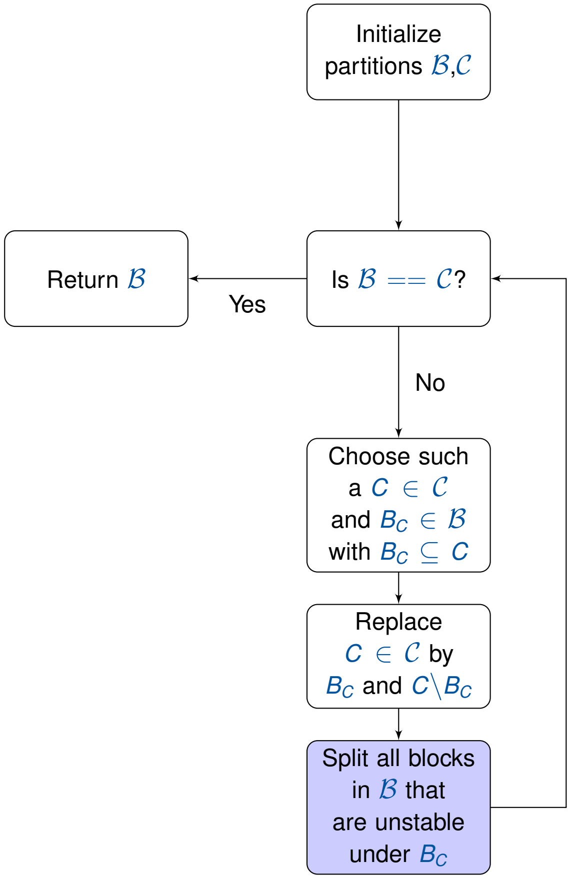
$$C = U$$

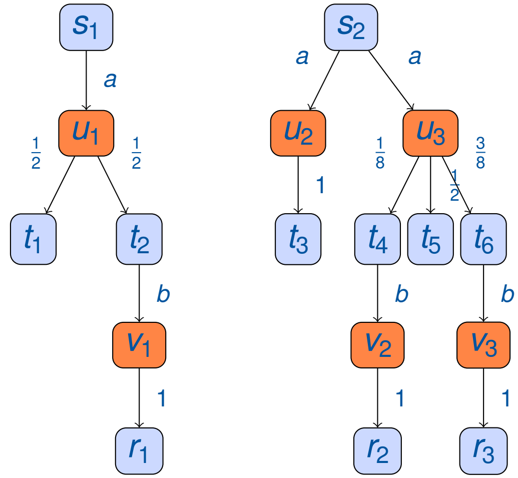
$$B_C = \{u_2, v_{1-3}\}$$

$$\mathcal{C} = \{\{u_1, u_3\}, \{u_2, v_{1-3}\}, \{s_1, s_2\}, \{t_2, t_4, t_6\}, \{t_1, t_3, t_5, r_{1-3}\}\}$$

$$\mathcal{B} = \{\{u_1, u_3\}, \{u_2, v_{1-3}\}, \{s_1, s_2\}, \{t_2, t_4, t_6\}, \{t_1, t_3, t_5, r_{1-3}\}\}$$

Algorithm





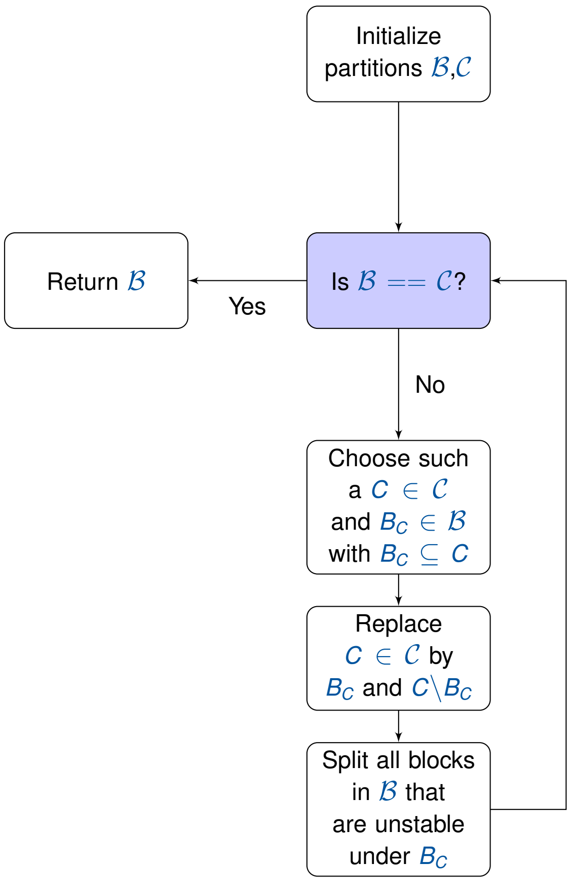
$$C = U$$

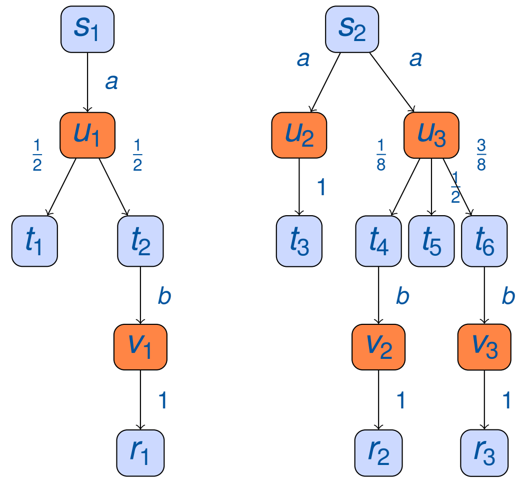
$$B_C = \{u_2, v_{1-3}\}$$

$$\mathcal{C} = \{\{u_1, u_3\}, \{u_2, v_{1-3}\}, \{s_1, s_2\}, \{t_2, t_4, t_6\}, \{t_1, t_3, t_5, r_{1-3}\}\}$$

$$\mathcal{B} = \{\{u_1, u_3\} \{u_2, v_{1-3}\}, \{s_1\}, \{s_2\}, \{t_2, t_4, t_6\}, \{t_1, t_3, t_5, r_{1-3}\}\}$$

Algorithm

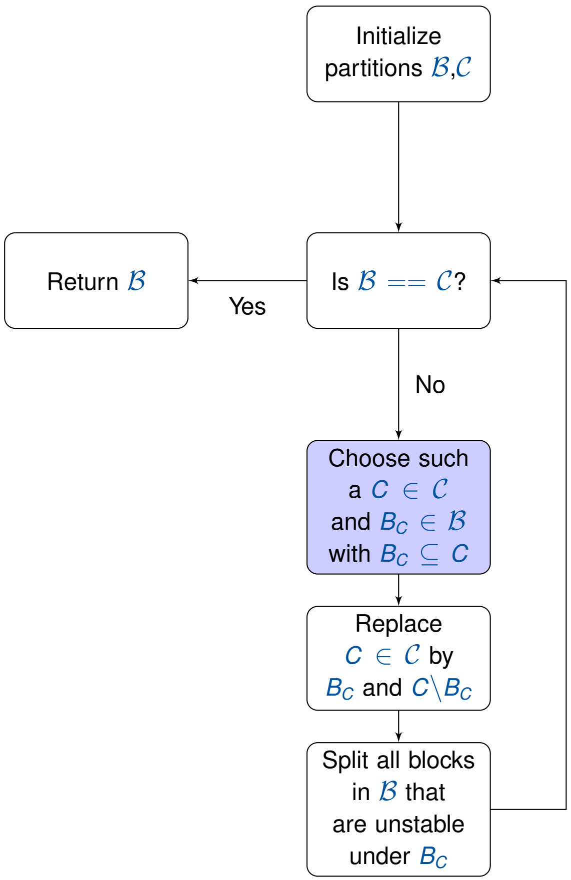


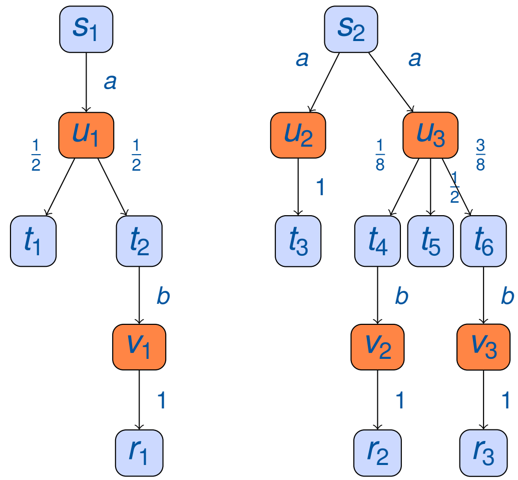


$$\mathcal{C} = \{\{u_1, u_3\}, \{u_2, v_{1-3}\}, \{s_1, s_2\}, \{t_2, t_4, t_6\}, \{t_1, t_3, t_5, r_{1-3}\}\}$$

$$\mathcal{B} = \{\{u_1, u_3\}\ \{u_2, v_{1-3}\}, \{s_1\}, \{s_2\}, \{t_2, t_4, t_6\}, \{t_1, t_3, t_5, r_{1-3}\}\}$$

Algorithm





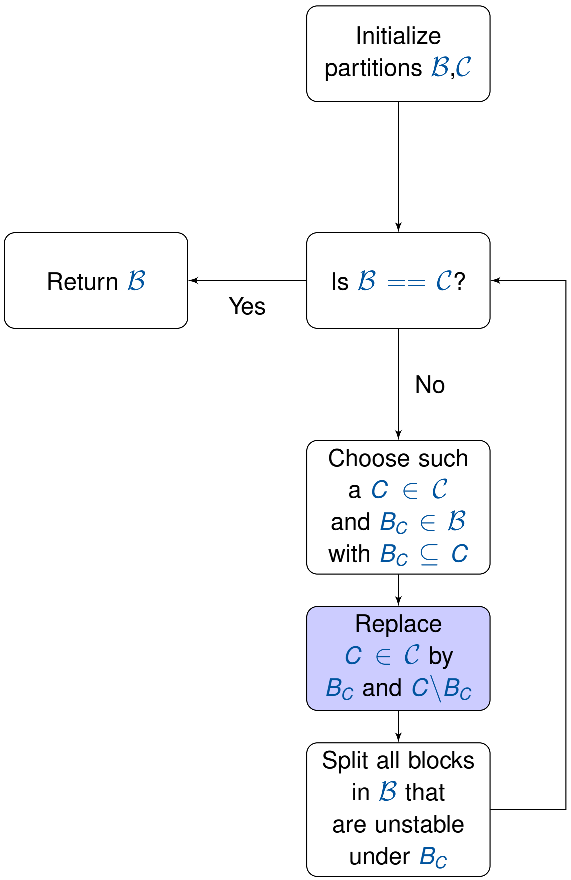
$$C = \{s_1, s_2\}$$

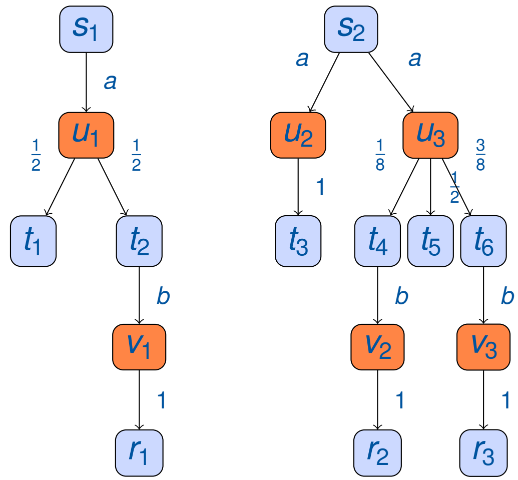
$$B_C = \{s_1\}$$

$$\mathcal{C} = \{\{u_1, u_3\}, \{u_2, v_{1-3}\}, \{s_1, s_2\}, \{t_2, t_4, t_6\}, \{t_1, t_3, t_5, r_{1-3}\}\}$$

$$\mathcal{B} = \{\{u_1, u_3\} \{u_2, v_{1-3}\}, \{s_1\}, \{s_2\}, \{t_2, t_4, t_6\}, \{t_1, t_3, t_5, r_{1-3}\}\}$$

Algorithm





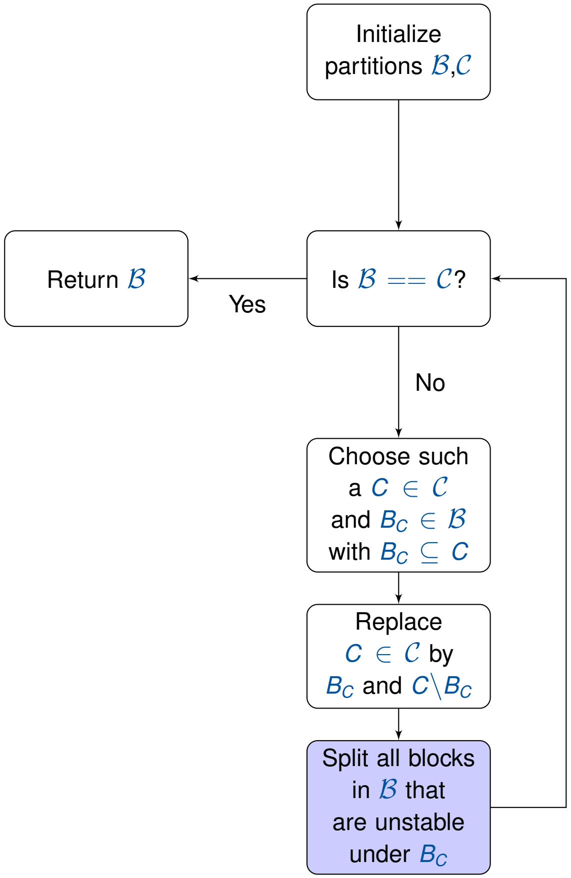
$$C = \{s_1, s_2\}$$

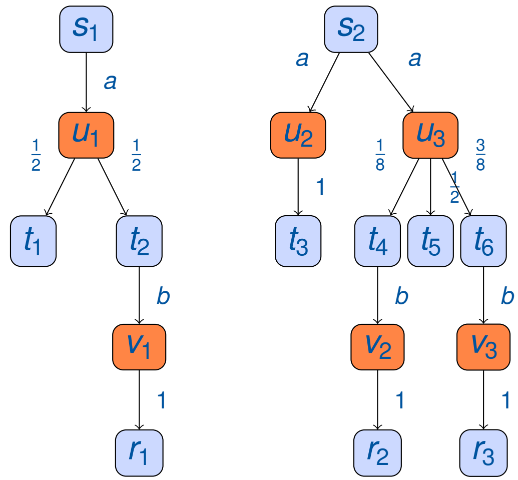
$$B_C = \{s_1\}$$

$$\mathcal{C} = \{\{u_1, u_3\}\,\{u_2, v_{1-3}\}\,, \{s_1\}, \{s_2\}, \{t_2, t_4, t_6\}\,, \{t_1, t_3, t_5, r_{1-3}\}\}$$

$$\mathcal{B} = \{\{u_1, u_3\}\,\{u_2, v_{1-3}\}\,, \{s_1\}, \{s_2\}, \{t_2, t_4, t_6\}\,, \{t_1, t_3, t_5, r_{1-3}\}\}$$

Algorithm





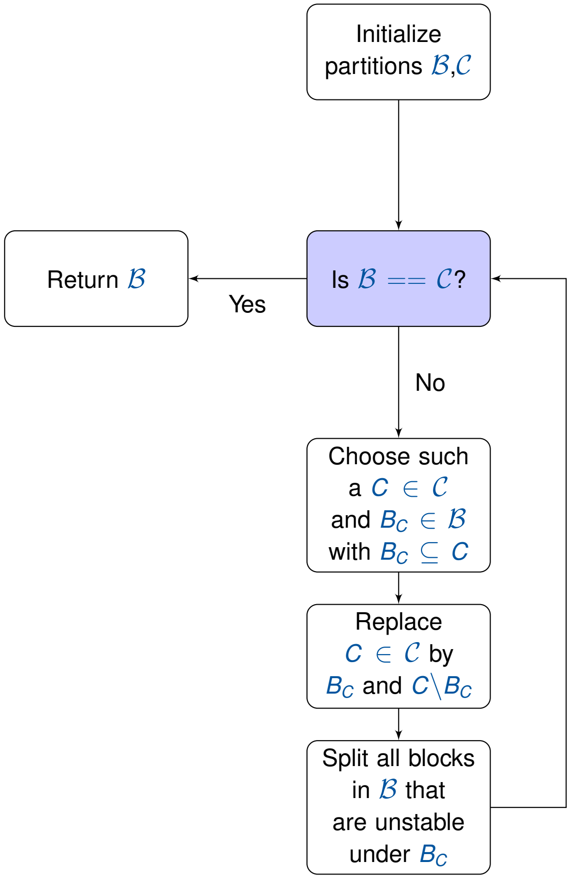
$$C = \{s_1, s_2\}$$

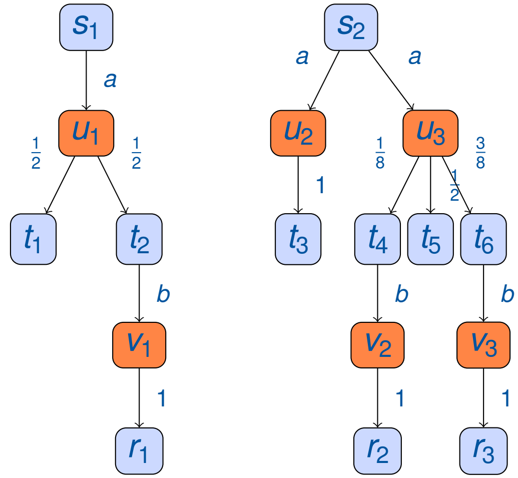
$$B_C = \{s_1\}$$

$$\mathcal{C} = \{\{u_1, u_3\}\,\{u_2, v_{1-3}\}, \{s_1\}, \{s_2\}, \{t_2, t_4, t_6\}, \{t_1, t_3, t_5, r_{1-3}\}\}$$

$$\mathcal{B} = \{\{u_1, u_3\}\,\{u_2, v_{1-3}\}, \{s_1\}, \{s_2\}, \{t_2, t_4, t_6\}, \{t_1, t_3, t_5, r_{1-3}\}\}$$

Algorithm





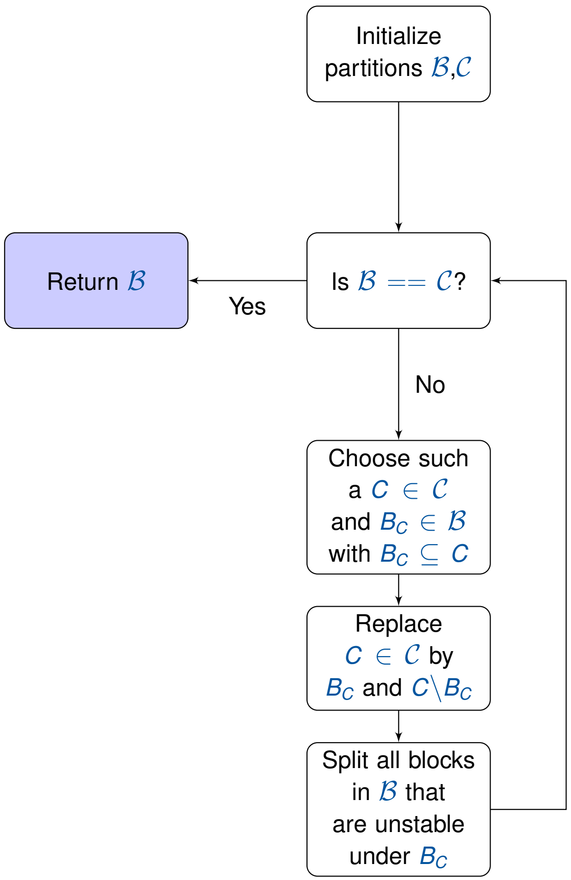
$$C = \{s_1, s_2\}$$

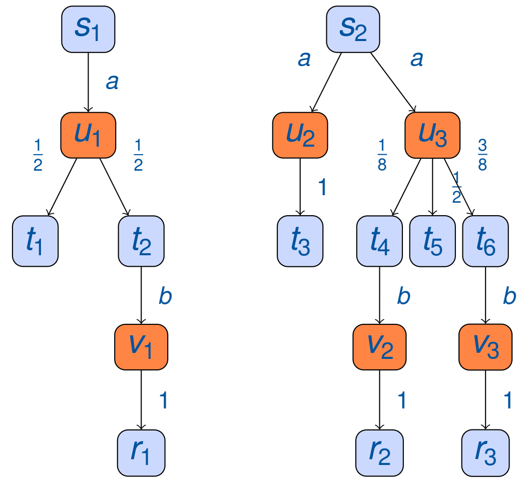
$$B_C = \{s_1\}$$

$$\mathcal{C} = \{\{u_1, u_3\}\,\{u_2, v_{1-3}\}, \{s_1\}, \{s_2\}, \{t_2, t_4, t_6\}, \{t_1, t_3, t_5, r_{1-3}\}\}$$

$$\mathcal{B} = \{\{u_1, u_3\}\,\{u_2, v_{1-3}\}, \{s_1\}, \{s_2\}, \{t_2, t_4, t_6\}, \{t_1, t_3, t_5, r_{1-3}\}\}$$

Algorithm





$$C = \{s_1, s_2\}$$

$$B_C = \{s_1\}$$

$$\mathcal{C} = \{\{u_1, u_3\}\{u_2, v_{1-3}\}, \{s_1\}, \{s_2\}, \{t_2, t_4, t_6\}, \{t_1, t_3, t_5, r_{1-3}\}\}$$

$$\mathcal{B} = \{\{u_1, u_3\}\{u_2, v_{1-3}\}, \{s_1\}, \{s_2\}, \{t_2, t_4, t_6\}, \{t_1, t_3, t_5, r_{1-3}\}\}$$

Outline

Software Modeling and Verification Chair

RWTH AACHEN UNIVERSITY

Outline

Correctness

Invariant 1

Probabilistic bisimilarity $\simeq_p$ is a refinement of $\mathcal{B}$.

Correctness

Invariant 1

Probabilistic bisimilarity $\simeq_p$ is a refinement of $\mathcal{B}$.

Invariant 2

Partition $\mathcal{B}$ is a refinement of partition $\mathcal{C}$.

Analysis

Correctness

Invariant 1

Probabilistic bisimilarity $\simeq_p$ is a refinement of $\mathcal{B}$.

Invariant 2

Partition $\mathcal{B}$ is a refinement of partition $\mathcal{C}$.

Invariant 3

Partition $\mathcal{B}$ is stable under the set of constellations $\mathcal{C}$.

Analysis

Correctness

Invariant 1

Probabilistic bisimilarity $\simeq_p$ is a refinement of $\mathcal{B}$.

Invariant 2

Partition $\mathcal{B}$ is a refinement of partition $\mathcal{C}$.

Invariant 3

Partition $\mathcal{B}$ is stable under the set of constellations $\mathcal{C}$.

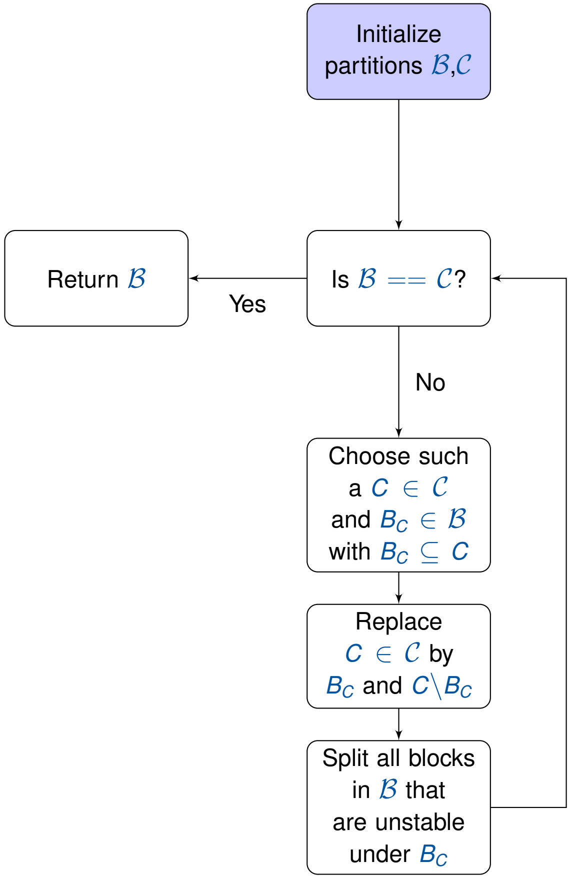
Lemma

If $\mathcal{B}$ is stable under itself, then $\sim_{\mathcal{B}}$ is a probabilistic bisimulation.

Outline

Software Modeling and Verification Chair
RWTH AACHEN UNIVERSITY

Complexity



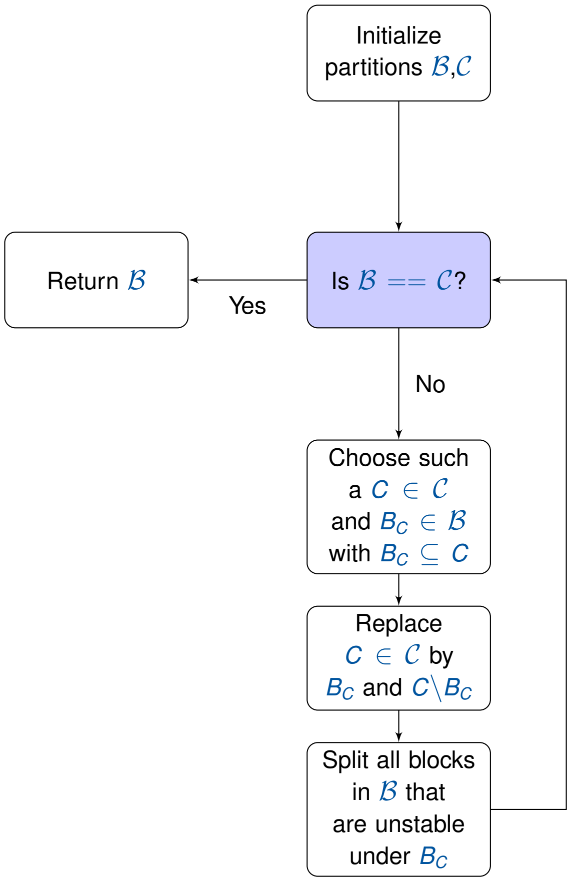
Notation

$n_a :=$ number of action states
$n_p :=$ number of probabilistic states
$m_a :=$ number of action transitions
$m_p :=$ number of probabilistic transitions

- Initialization: $O(1)$

Analysis

Complexity

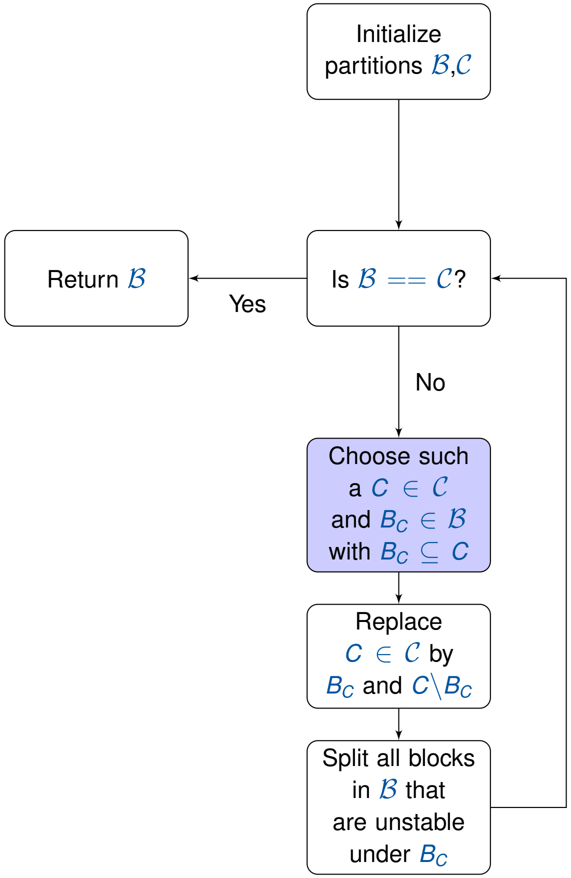


Notation

$n_a :=$ number of action states
$n_p :=$ number of probabilistic states
$m_a :=$ number of action transitions
$m_p :=$ number of probabilistic transitions

- Initialization: $O(1)$
- Checking invariant: $O(1)$

Complexity

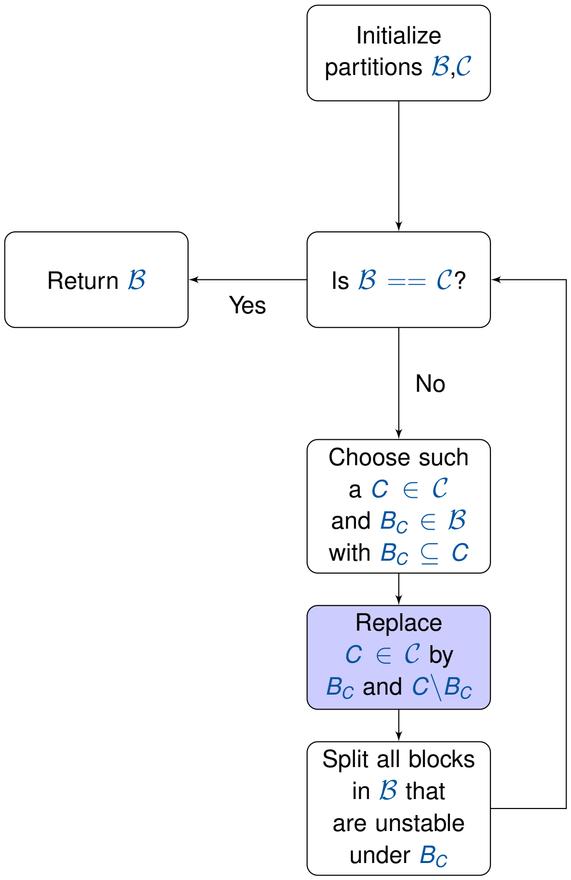


Notation

$n_a :=$ number of action states
$n_p :=$ number of probabilistic states
$m_a :=$ number of action transitions
$m_p :=$ number of probabilistic transitions

- Initialization: $O(1)$
- Checking invariant: $O(1)$
- Choosing appropriate subsets: $O(1)$
  1. Keep a stack of constellations that need to be checked
  2. Choose $|B_C| \leq \frac{1}{2}|C|$
  3. Number of iterations $\leq \log(n_a) + \log(n_p)$

Complexity

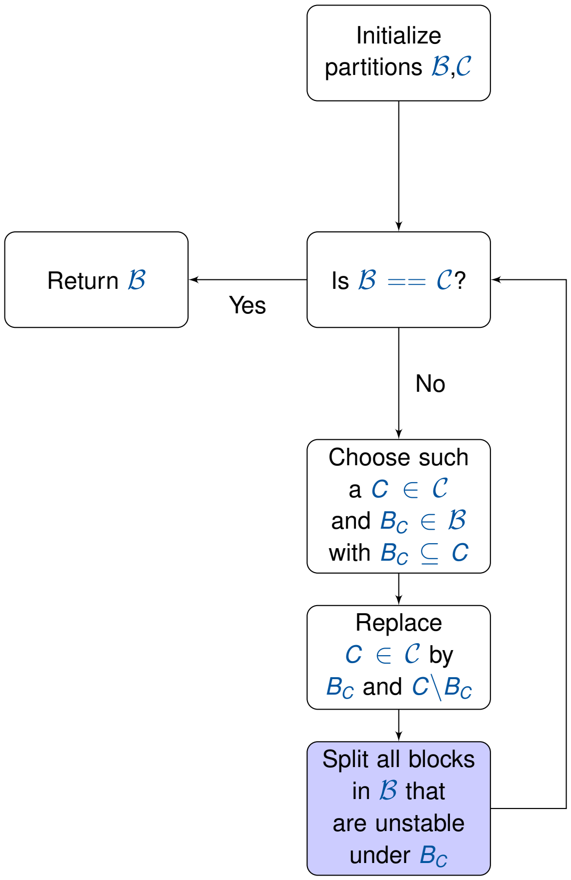


Notation

$n_a :=$ number of action states
$n_p :=$ number of probabilistic states
$m_a :=$ number of action transitions
$m_p :=$ number of probabilistic transitions

- Initialization: $O(1)$
- Checking invariant: $O(1)$
- Choosing appropriate subsets: $O(1)$
  1. Keep a stack of constellations that need to be checked
  2. Choose $|B_C| \leq \frac{1}{2}|C|$
  3. Number of iterations $\leq \log(n_a) + \log(n_p)$
- Refining $\mathcal{C}$: $O(1)$

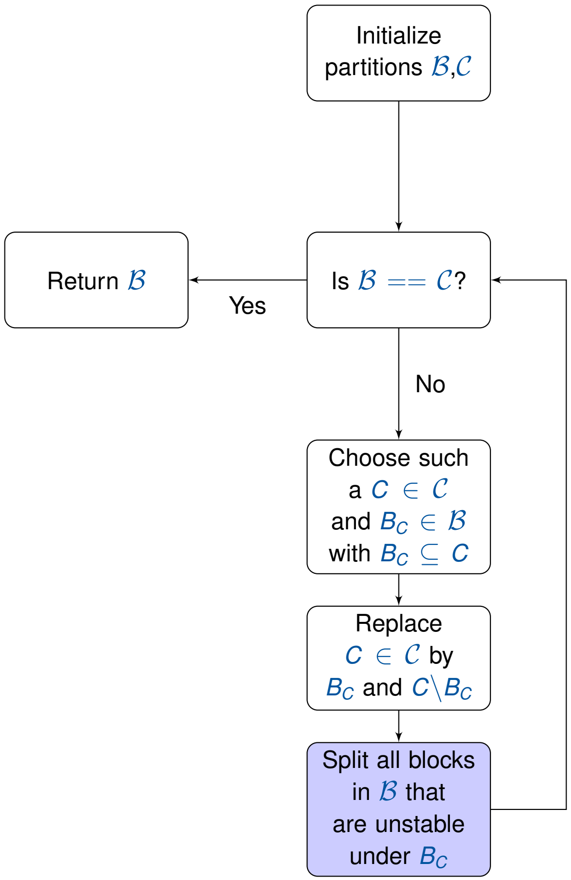
Complexity



- Refining $\mathcal{B}$:
  1. Mark unstable blocks B and determine the subsets B' that it has to be split into
  2. Create a new block for each B' for which $|B'| \leq \frac{1}{2}|B|$
  3. Each action state will only be moved $O(\log(n_a))$ times
  4. Each prob. state will only be moved $O(\log(n_p))$ times

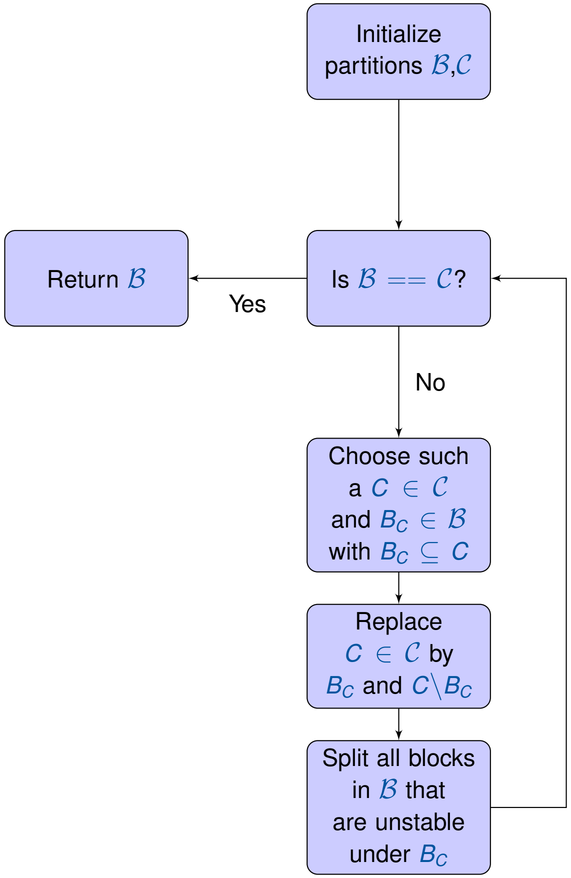
Analysis

Complexity



- Refining $\mathcal{B}$:
  1. Mark unstable blocks B and determine the subsets B' that it has to be split into
  2. Create a new block for each B' for which $|B'| \leq \frac{1}{2}|B|$
  3. Each action state will only be moved $O(\log(n_a))$ times
  4. Each prob. state will only be moved $O(\log(n_p))$ times
- Marking unstable blocks:
  1. This mainly works by iterating over incoming transitions to a block
  2. Marking procedures have complexity $O(\text{number of incoming transitions into } B_C)$
  3. Requires sorting that accumulates to $O(m_p \log(n_p))$

Complexity



Accumulated time complexity

In case all action states are reachable, the time complexity of the algorithm is $O((m_a + m_p) \log n_p + m_p \log n_a))$

Space complexity

All data structures are linear in the number of transitions and states. Thus, space complexity is $O\left(m_a + m_p\right)$

Outline

Outlook

Conclusion

The algorithm from Groote, Verduzco and de Vink is currently an efficient algorithm for computing probabilistic bisimilarity. It outperforms all previously known approaches.

Open research questions

- What is the best strategy of choosing blocks B_C? Right now, they are chosen non-deterministically
- Can this partition refinement approach be generalized for notions of equivalence other than bisimulation?

Outlook

References I

- C. Baier, B. Engelen, and M. Majster-Cederbaum.
  Deciding bisimilarity and similarity for probabilistic processes.
  Journal of Computer and System Sciences, 60(1):187 – 231, 2000.

- C. Baier and J.-P. Katoen.
  Principles of Model Checking.
  The MIT Press, 2008.

- J. Groote, H. Rivera Verduzco, and E. de Vink.
  An efficient algorithm to determine probabilistic bisimulation.
  Algorithms, 11(9):131, Sep 2018.

- R. Paige and R. E. Tarjan.
  Three partition refinement algorithms.
  SIAM Journal on Computing, 16(6):973–989, 1987.

Outlook

References II

- R. Segala and N. Lynch.
  Probabilistic simulations for probabilistic processes.
  In B. Jonsson and J. Parrow, editors, *CONCUR '94: Concurrency Theory*, pages 481–496, Berlin, Heidelberg, 1994. Springer Berlin Heidelberg.
- A. Turrini and H. Hermanns.
  Polynomial time decision algorithms for probabilistic automata.
  Information and Computation, 244:134 – 171, 2015.
- A. Valmari and G. Franceschinis.
  Simple o(m logn) time markov chain lumping.
  In J. Esparza and R. Majumdar, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, pages 38–52, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

Outline

Backup slides

Quotient transition system

For transition system $TS = (S, Act, \rightarrow, I, AP, L)$ and bisimulation $\sim_{TS}$, the quotient transition system $TS/\sim_{TS}$ is defined by

$$TS/\sim_{TS} = (S/\sim_{TS}, \tau, \longrightarrow', I', AP, L')$$

where:

- $I' = \{[s]_{\sim_{TS}} | \in I\}$
- $\rightarrow'$ is defined by

$$\frac{s \xrightarrow{\alpha} s'}{[s]_{\sim_{TS}} \xrightarrow{\tau} [s']_{\sim_{TS}}}$$

- $L'([s]_{\sim_{TS}}) = L(s)$

Bijection between equivalence relations and partitions

Let X be a finite set, $P \subseteq \mathcal{P}(X)$ a partition of X and $\sim$ an equivalence relation on X.

1. P induces an equivalence relation $\sim_P$ defined by $x \sim_P y \iff \exists A \in P : x \in A \land y \in A$
2. $\sim$ induces a partition $P_\sim := \{[x]_\sim \mid x \in X\}$ where $[x]_\sim$ denotes the equivalence class of x
3. There is a bijection between the set of equivalence relations on X and the set of partitions of X

Stability under itself

Lemma

Let $A = (S, \rightarrow)$ be a PLTS and $\mathcal{B}$ a partition of S.
If $\mathcal{B}$ is stable under itself, then the corresponding equivalence relation $\sim_B$ is a probabilistic bisimulation.

Proof

We check the definition of probabilistic bisimilarity:
Suppose two action states $s, t \in S$ are in the same equivalence class $B \in \mathcal{B}$. Let there be a transition $s \xrightarrow{a} u_f$ with $f \in \mathcal{D}(S)$. Then there is a block $B' \in P$ which contains u_f, so that $s \xrightarrow{a} B'$. Since $\mathcal{B}$ is stable under itself, B is stable under B'. Consequently, there is $g \in B'$ such that $t \xrightarrow{a} u_g$.
Now let $B'' \in P$ be an arbitrary block. Since B' is stable under B'', it holds that

$$u_f[B''] = u_g[B'']$$

Complete algorithm

Algorithm 2 Partition refinement algorithm for probabilistic bisimulation

function Partition Refinement $(S, U, \rightarrow)$	
$\mathcal{C} := \{S, U\}$	$O(n_a + n_p)$
$\mathcal{B} := \{U\} \cup \{S_A \mid A \subseteq Act\}$ where $S_A = \left\{ s \in S \mid \forall a \in Act \left(\exists u \in U : s \xrightarrow{a} u \iff a \in A \right) \right\}$	$O(n_p + n_a + m_a)$
group the incoming action transitions in each block per label	$O(m_a)$
initialise `state_to_constellation_cht` for each transition	$O(m_a)$
while $\mathcal{C}$ contains a non-trivial constellation C **do**	$\leq n$ iterations
choose a block B_C from $\mathcal{B}$ in C such that $\lvert B_C \rvert \leq \frac{1}{2}\lvert C \rvert$	
split constellation C into B_C and $C \backslash B_C$ in $\mathcal{C}$	$O(1)$
if C contains probabilistic states **then**	
for all incoming actions a of states in B_C **do**	$\leq \lvert Act \rvert$ iterations
$\langle \mathbf{B}_a, \text{left}_a, \text{mid}_a, \text{right}_a, \text{large}_a \rangle :=$aMark$(\mathcal{B}, C, B_C, a)$	$O($ nr of incoming a transitions in $B_C)$
for all blocks $B \in \mathbf{B}_a$ **do**	
for all non-empty subsets $B' \subseteq B$, different from $\text{large}_a(B)$ in $\{\text{left}(B)_a, \text{mid}_a(B), \text{right}_a(B)\}$ **do**	$O($ nr of incoming a transitions in $B_C)$
move B' out of B and add B' as new block to $\mathcal{B}$	$O($ nr of incoming transitions in $B')$
else	$O(\text{nr of incoming prob. transitions in } B_C)$ plus
$\langle \mathbf{B}_p, \text{left}_p, \text{mid}_p, \text{right}_p, \text{large}_p \rangle :=$ pMark $(\mathcal{B}, C, B_C)$	a sorting penalty
for all blocks $B \in \mathbf{B}_p$ **do**	
for all non-empty sets of states $B' \subseteq B$ not equal to $\text{large}_p(B)$ in $\{\text{left}_p(B)\} \cup \text{mid}_p(B) \cup \{\text{right}(B)_p\}$ **do**	$O($ nr of incoming prob. transitions in $B_C)$
move B' out of B and add B' as a new block to $\mathcal{B}$	$O($ nr of incoming transitions in $B')$
return $\mathcal{B}$	