# Overview



Introduction

Modelling parallel systems

Linear Time Properties

Regular Properties

Linear Temporal Logic (LTL)

Computation-Tree Logic

**Equivalences and Abstraction**

- bisimulation
- CTL, CTL*-equivalence
- computing the bisimulation quotient ⟵
- abstraction stutter steps
- simulation relations

# Bisimulation quotient $\mathcal{T}/\!\sim$



$\mathcal{T}/\!\sim$ arises by collapsing all bisimilar states in $\mathcal{T}$

# Bisimulation quotient $\mathcal{T}/\sim$



$\mathcal{T}/\sim$ arises by collapsing all bisimilar states in $\mathcal{T}$

- *states* of $\mathcal{T}/\sim$: bisimulation equivalence classes of $\mathcal{T}$

# Bisimulation quotient $\mathcal{T}/\!\sim$



$\mathcal{T}/\!\sim$ arises by collapsing all bisimilar states in $\mathcal{T}$

- *states* of $\mathcal{T}/\!\sim$: bisimulation equivalence classes of $\mathcal{T}$
- *transitions:* arise by lifting $\mathcal{T}$'s transitions to the bisimulation equivalence classes

# Bisimulation quotient $\mathcal{T}/\!\sim$

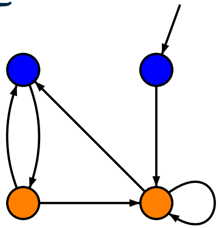


$\mathcal{T}/\!\sim$ arises by collapsing all bisimilar states in $\mathcal{T}$

- *states* of $\mathcal{T}/\!\sim$: bisimulation equivalence classes of $\mathcal{T}$
- *transitions:* arise by lifting $\mathcal{T}$'s transitions to the bisimulation equivalence classes

---

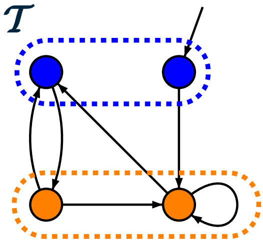
$\mathcal{T}$



$\mathcal{T}/\sim$ arises by collapsing all bisimilar states in $\mathcal{T}$

- *states* of $\mathcal{T}/\sim$: bisimulation equivalence classes of $\mathcal{T}$
- *transitions:* arise by lifting $\mathcal{T}$'s transitions to the bisimulation equivalence classes

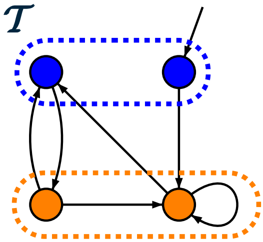
---

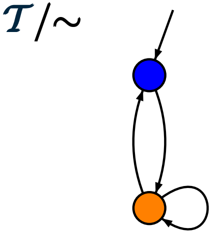


# Bisimulation quotient $\mathcal{T}/\!\sim$

$\mathcal{T}/\!\sim$ arises by collapsing all bisimilar states in $\mathcal{T}$

- *states* of $\mathcal{T}/\!\sim$: bisimulation equivalence classes of $\mathcal{T}$
- *transitions:* arise by lifting $\mathcal{T}$'s transitions to the bisimulation equivalence classes



# Applications of the bisimulation quotient

1. equivalence checking: check whether $\mathcal{T}_1 \sim \mathcal{T}_2$

# Applications of the bisimulation quotient

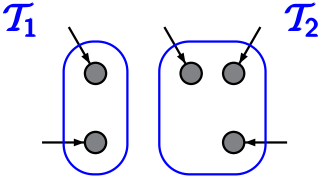


1. equivalence checking: check whether $\mathcal{T}_1 \sim \mathcal{T}_2$
   for two transition systems $\mathcal{T}_1$, $\mathcal{T}_2$,
   e.g., abstract model and its refinement

# Applications of the bisimulation quotient



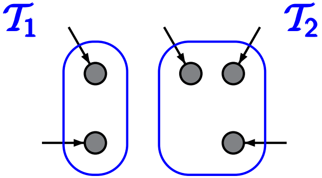
1. equivalence checking: check whether $\mathcal{T}_1 \sim \mathcal{T}_2$
   for two transition systems $\mathcal{T}_1$, $\mathcal{T}_2$,
   e.g., abstract model and its refinement

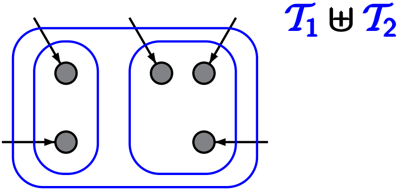


# Applications of the bisimulation quotient



1. equivalence checking: check whether $\mathcal{T}_1 \sim \mathcal{T}_2$
   for two transition systems $\mathcal{T}_1$, $\mathcal{T}_2$,
   e.g., abstract model and its refinement
   regard $\mathcal{T}_1 \uplus \mathcal{T}_2$



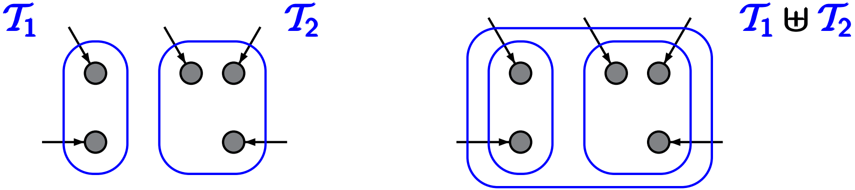
# Applications of the bisimulation quotient 

1. equivalence checking: check whether $\mathcal{T}_1 \sim \mathcal{T}_2$
   for two transition systems $\mathcal{T}_1$, $\mathcal{T}_2$,
   e.g., abstract model and its refinement

   regard $\mathcal{T}_1 \uplus \mathcal{T}_2$ and check whether for all bisimulation equivalence classes $C$ in $\mathcal{T}_1 \uplus \mathcal{T}_2$:

$$C \cap S_{0,1} \neq \varnothing \quad \text{iff} \quad C \cap S_{0,2} \neq \varnothing$$

   where $S_{0,i}$ is the set of initial states in $\mathcal{T}_i$



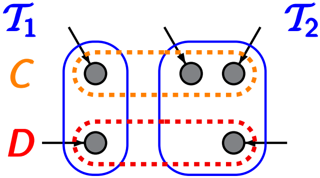
1. equivalence checking: check whether $\mathcal{T}_1 \sim \mathcal{T}_2$

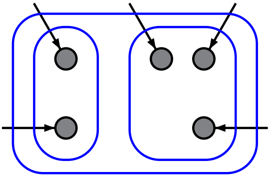
   for two transition systems $\mathcal{T}_1$, $\mathcal{T}_2$,
   e.g., abstract model and its refinement

   regard $\mathcal{T}_1 \uplus \mathcal{T}_2$ and check whether for all bisimulation equivalence classes $C$ in $\mathcal{T}_1 \uplus \mathcal{T}_2$:

$$C \cap S_{0,1} \neq \varnothing \quad \text{iff} \quad C \cap S_{0,2} \neq \varnothing$$

   where $S_{0,i}$ is the set of initial states in $\mathcal{T}_i$



# Applications of the bisimulation quotient



1. equivalence checking: check whether $\mathcal{T}_1 \sim \mathcal{T}_2$
   for two transition systems $\mathcal{T}_1$, $\mathcal{T}_2$,
   e.g., abstract model and its refinement

   regard $\mathcal{T}_1 \uplus \mathcal{T}_2$ and check whether for all bisimulation equivalence classes $C$ in $\mathcal{T}_1 \uplus \mathcal{T}_2$:

   $$C \cap S_{0,1} \neq \varnothing \quad \text{iff} \quad C \cap S_{0,2} \neq \varnothing$$

   where $S_{0,i}$ is the set of initial states in $\mathcal{T}_i$

2. graph minimization:

# Applications of the bisimulation quotient



1. equivalence checking: check whether $\mathcal{T}_1 \sim \mathcal{T}_2$

   for two transition systems $\mathcal{T}_1$, $\mathcal{T}_2$,
   e.g., abstract model and its refinement

   regard $\mathcal{T}_1 \uplus \mathcal{T}_2$ and check whether for all bisimulation equivalence classes $C$ in $\mathcal{T}_1 \uplus \mathcal{T}_2$:

   $$C \cap S_{0,1} \neq \varnothing \quad \text{iff} \quad C \cap S_{0,2} \neq \varnothing$$

   where $S_{0,i}$ is the set of initial states in $\mathcal{T}_i$

2. graph minimization:

   replace $\mathcal{T}$ with $\mathcal{T}/{\sim}$ and analyze $\mathcal{T}/{\sim}$

# Computing the bisimulation quotient



.... relies on a partitioning refinement algorithm ...

# Computing the bisimulation quotient



.... relies on a partitioning refinement algorithm ...

*here*: only explanations for finite transition systems, possibly with terminal states

# Notations: partitions and co.

# Notations: partitions and co.

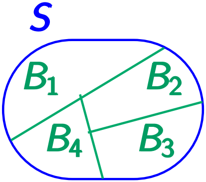


$\mathcal{T} = (S, Act, \rightarrow, S_0, AP, L)$ finite transition system

# Notations: partitions and co.

$\mathcal{T} = (S, Act, \rightarrow, S_0, AP, L)$ finite transition system

*partition* for $\mathcal{T}$: decomposition of the state space $S$ into pairwise disjoint nonempty subsets

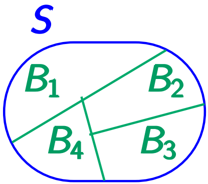



$$\mathcal{B} = \{B_1, \ldots, B_k\}$$

# Notations: partitions, block

$\mathcal{T} = (S, Act, \rightarrow, S_0, AP, L)$ finite transition system

*partition* for $\mathcal{T}$: decomposition of the state space $S$ into pairwise disjoint nonempty subsets




$\mathcal{B} = \{B_1, \ldots, B_k\}$ s.t.

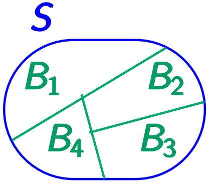
- $B_i \neq \varnothing$
- $B_i \cap B_j = \varnothing$ for $i \neq j$
- $S = B_1 \cup \ldots \cup B_k$

The $B_i$'s are called blocks of $\mathcal{B}$.

# Notations: partitions, block, superblock

$\mathcal{T} = (S, Act, \to, S_0, AP, L)$ finite transition system

*partition* for $\mathcal{T}$: decomposition of the state space $S$ into pairwise disjoint nonempty subsets




$\mathcal{B} = \{B_1, \ldots, B_k\}$ s.t.

- $B_i \neq \varnothing$
- $B_i \cap B_j = \varnothing$ for $i \neq j$
- $S = B_1 \cup \ldots \cup B_k$

The $B_i$'s are called blocks of $\mathcal{B}$.
A superblock denotes any union of blocks.

# Partitions and equivalences



$$\boxed{\text{partitions} \quad \widehat{=} \quad \text{equivalences on } S}$$

partitions $\mathrel{\widehat{=}}$ equivalences on $S$

- partition $\mathcal{B} \rightsquigarrow$ equivalence relation $\mathcal{R}_{\mathcal{B}}$ where

$$\mathcal{R}_{\mathcal{B}} = \big\{(s, s') : [s]_{\mathcal{B}} = [s']_{\mathcal{B}}\big\}$$

# Partitions and equivalences

$$\boxed{\text{partitions} \quad \widehat{=} \quad \text{equivalences on } S}$$

- partition $\mathcal{B} \rightsquigarrow$ equivalence relation $\mathcal{R}_{\mathcal{B}}$ where

$$\mathcal{R}_{\mathcal{B}} = \big\{(s, s') : [s]_{\mathcal{B}} = [s']_{\mathcal{B}}\big\}$$

$$[s]_{\mathcal{B}} = \text{unique block } B_i \in \mathcal{B} \text{ with } s \in B_i$$

# Partitions and equivalences



$$\boxed{\text{partitions} \quad \widehat{=} \quad \text{equivalences on } S}$$

- partition $\mathcal{B} \rightsquigarrow$ equivalence relation $\mathcal{R}_{\mathcal{B}}$ where

  $$\mathcal{R}_{\mathcal{B}} = \{(s, s') : [s]_{\mathcal{B}} = [s']_{\mathcal{B}}\}$$

  $$[s]_{\mathcal{B}} = \text{unique block } B_i \in \mathcal{B} \text{ with } s \in B_i$$

- equivalence $\mathcal{R}$ on $S \rightsquigarrow$ partition $\mathcal{B} = S/\mathcal{R}$

# Notations for partitions: finer, coarser

# Notations for partitions: finer, coarser



Let $\mathcal{B}_1$ and $\mathcal{B}_2$ be partitions for $\mathcal{T}$.

# Notations for partitions: finer, coarser



Let $\mathcal{B}_1$ and $\mathcal{B}_2$ be partitions for $\mathcal{T}$.

$\mathcal{B}_1$ is called *finer* than $\mathcal{B}_2$ (and $\mathcal{B}_2$ *coarser* than $\mathcal{B}_1$) if

$$\forall B \in \mathcal{B}_1 \; \exists B' \in \mathcal{B}_2 \text{ such that } B \subseteq B'$$

Let $\mathcal{B}_1$ and $\mathcal{B}_2$ be partitions for $\mathcal{T}$.

$\mathcal{B}_1$ is called *finer* than $\mathcal{B}_2$ (and $\mathcal{B}_2$ *coarser* than $\mathcal{B}_1$) if

$$\forall B \in \mathcal{B}_1 \; \exists B' \in \mathcal{B}_2 \text{ such that } B \subseteq B',$$

i.e., if all blocks $B' \in \mathcal{B}_2$ are superblocks of $\mathcal{B}_1$

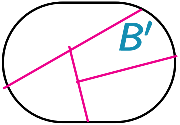
# Notations for partitions: finer, coarser

Let $\mathcal{B}_1$ and $\mathcal{B}_2$ be partitions for $\mathcal{T}$.

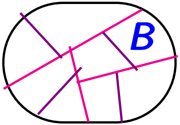
$\mathcal{B}_1$ is called *finer* than $\mathcal{B}_2$ (and $\mathcal{B}_2$ *coarser* than $\mathcal{B}_1$) if

$$\forall B \in \mathcal{B}_1 \ \exists B' \in \mathcal{B}_2 \text{ such that } B \subseteq B',$$

i.e., if all blocks $B' \in \mathcal{B}_2$ are superblocks of $\mathcal{B}_1$




$\mathcal{B}_2$




$\mathcal{B}_1$

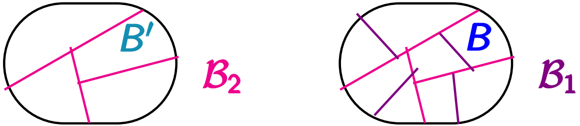
# Notations for partitions: finer, coarser

Let $\mathcal{B}_1$ and $\mathcal{B}_2$ be partitions for $\mathcal{T}$.

$\mathcal{B}_1$ is called *finer* than $\mathcal{B}_2$ (and $\mathcal{B}_2$ *coarser* than $\mathcal{B}_1$) if

$$\forall B \in \mathcal{B}_1 \; \exists B' \in \mathcal{B}_2 \text{ such that } B \subseteq B',$$

i.e., if all blocks $B' \in \mathcal{B}_2$ are superblocks of $\mathcal{B}_1$



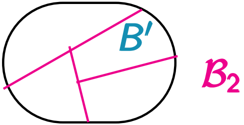

*Example:* if $\mathcal{R}$ is a bisimulation for $\mathcal{T}$ and an equivalence then $S/\mathcal{R}$ is *finer* than $S/\sim$

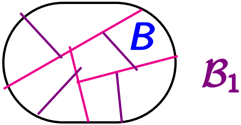
Let $\mathcal{B}_1$ and $\mathcal{B}_2$ be partitions for $\mathcal{T}$.

$\mathcal{B}_1$ is called *finer* than $\mathcal{B}_2$ (and $\mathcal{B}_2$ *coarser* than $\mathcal{B}_1$) if

$$\forall B \in \mathcal{B}_1 \; \exists B' \in \mathcal{B}_2 \text{ such that } B \subseteq B',$$

i.e., if all blocks $B' \in \mathcal{B}_2$ are superblocks of $\mathcal{B}_1$




$\mathcal{B}_1$ is called *strictly finer* than $\mathcal{B}_2$ if

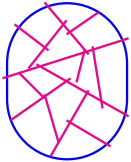
(1) $\mathcal{B}_1$ is finer than $\mathcal{B}_2$ and (2) $\mathcal{B}_1 \neq \mathcal{B}_2$

# Computing the bisimulation quotient



by stepwise refinement of partitions the state set $S$

by stepwise refinement of partitions the state set $S$

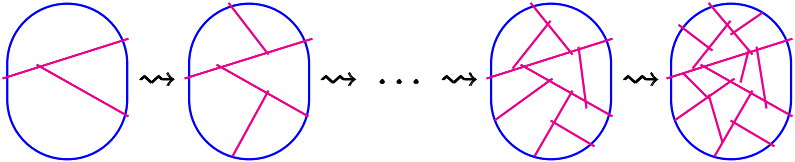


$S/\!\sim$

# Computing the bisimulation quotient



by stepwise refinement of partitions the state set $S$

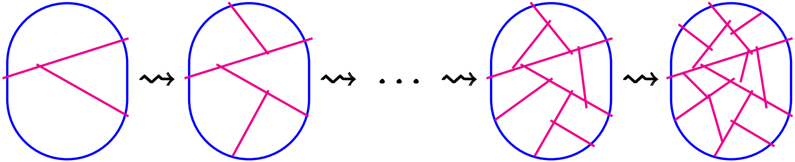


$\mathcal{B}_0 \rightsquigarrow \mathcal{B}_1 \rightsquigarrow \ldots \rightsquigarrow \mathcal{B}_{k-1} \rightsquigarrow \mathcal{B}_k = S/\sim$

# Computing the bisimulation quotient



by stepwise refinement of partitions the state set $S$



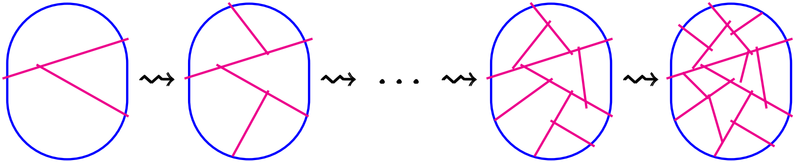
$\mathcal{B}_{AP} = \mathcal{B}_0$ $\quad \mathcal{B}_1$ $\quad \mathcal{B}_{k-1}$ $\quad \mathcal{B}_k = S/\!\sim$

initial partition: $\mathcal{B}_{AP} = \mathcal{B}_0$

identifies all states with the same labeling

# Computing the bisimulation quotient

by stepwise refinement of partitions the state set $S$



$\mathcal{B}_{AP} = \mathcal{B}_0 \quad \mathcal{B}_1 \quad \mathcal{B}_{k-1} \quad \mathcal{B}_k = S/\sim$

initial partition: $\mathcal{B}_{AP} = \mathcal{B}_0 = S/\mathcal{R}_{AP}$ where

$$\mathcal{R}_{AP} = \{ (s_1, s_2) : L(s_1) = L(s_2) \}$$

# Characterization of $S/{\sim_{\mathcal{T}}}$



... as the coarsest partition of the
state space $S$ such that ....

# Bisimulation equivalence $\sim_{\mathcal{T}}$



$\sim_{\mathcal{T}}$ is the coarsest equivalence on $S$ s.t.

# Bisimulation equivalence $\sim_{\mathcal{T}}$

$\sim_{\mathcal{T}}$ is the coarsest equivalence on $S$ s.t.

1. $s_1 \sim_{\mathcal{T}} s_2$ implies $L(s_1) = L(s_2)$
2. $\begin{array}{ccc} s_1 & \sim_{\mathcal{T}} & s_2 \\ \downarrow & & \\ s_1' & & \end{array}$ can be completed to $\begin{array}{ccc} s_1 & \sim_{\mathcal{T}} & s_2 \\ \downarrow & & \downarrow \\ s_1' & \sim_{\mathcal{T}} & s_2' \end{array}$

# Bisimulation quotient $S/{\sim_\mathcal{T}}$



$\sim_\mathcal{T}$ is the coarsest equivalence on $S$ s.t.

1. $s_1 \sim_\mathcal{T} s_2$ implies $L(s_1) = L(s_2)$
2. $\begin{array}{ccc} s_1 & \sim_\mathcal{T} & s_2 \\ \downarrow & & \\ s_1' & & \end{array}$ can be completed to $\begin{array}{ccc} s_1 & \sim_\mathcal{T} & s_2 \\ \downarrow & & \downarrow \\ s_1' & \sim_\mathcal{T} & s_2' \end{array}$

bisimulation quotient space $S/{\sim_\mathcal{T}}$:

coarsest partition $\mathcal{B}$ of the state space $S$ s.t.

# Bisimulation quotient $S/\sim_{\mathcal{T}}$

$\sim_{\mathcal{T}}$ is the coarsest equivalence on $S$ s.t.

1. $s_1 \sim_{\mathcal{T}} s_2$ implies $L(s_1) = L(s_2)$
2. $\begin{array}{ccc} s_1 & \sim_{\mathcal{T}} & s_2 \\ \downarrow & & \\ s_1' & & \end{array}$ can be completed to $\begin{array}{ccc} s_1 & \sim_{\mathcal{T}} & s_2 \\ \downarrow & & \downarrow \\ s_1' & \sim_{\mathcal{T}} & s_2' \end{array}$

bisimulation quotient space $S/\sim_{\mathcal{T}}$:
coarsest partition $\mathcal{B}$ of the state space $S$ s.t.

1. $\mathcal{B}$ is finer than $\mathcal{B}_{AP}$

$\sim_{\mathcal{T}}$ is the coarsest equivalence on $S$ s.t.

1. $s_1 \sim_{\mathcal{T}} s_2$ implies $L(s_1) = L(s_2)$
2. $\begin{array}{ccc} s_1 & \sim_{\mathcal{T}} & s_2 \\ \downarrow & & \\ s_1' & & \end{array}$ can be completed to $\begin{array}{ccc} s_1 & \sim_{\mathcal{T}} & s_2 \\ \downarrow & & \downarrow \\ s_1' & \sim_{\mathcal{T}} & s_2' \end{array}$

bisimulation quotient space $S/\!\sim_{\mathcal{T}}$:

coarsest partition $\mathcal{B}$ of the state space $S$ s.t.

1. $\mathcal{B}$ is finer than $\mathcal{B}_{AP}$
2. for all blocks $B, C \in \mathcal{B}$:

$$B \subseteq Pre(C) \quad \text{or} \quad B \cap Pre(C) = \varnothing$$

# Bisimulation quotient $S/{\sim_{\mathcal{T}}}$

$\sim_{\mathcal{T}}$ is the coarsest equivalence on $S$ s.t.

1. $s_1 \sim_{\mathcal{T}} s_2$ implies $L(s_1) = L(s_2)$
2. $\begin{array}{ccc} s_1 & \sim_{\mathcal{T}} & s_2 \\ \downarrow & & \\ s_1' & & \end{array}$ can be completed to $\begin{array}{ccc} s_1 & \sim_{\mathcal{T}} & s_2 \\ \downarrow & & \downarrow \\ s_1' & \sim_{\mathcal{T}} & s_2' \end{array}$

bisimulation quotient space $S/{\sim_{\mathcal{T}}}$:

coarsest partition $\mathcal{B}$ of the state space $S$ s.t.

1. $\mathcal{B}$ is finer than $\mathcal{B}_{AP}$
2. for all blocks $B, C \in \mathcal{B}$:

$$B \subseteq Pre(C) \quad \text{or} \quad B \cap Pre(C) = \varnothing$$

where $Pre(C) = \{ s \in S : \exists s' \in C \text{ s.t. } s \to s' \}$

# Partitioning refinement algorithm



*input:* finite TS $\mathcal{T}$ with state space $S$ over $AP$ (possibly with terminal states)

*output:* bisimulation quotient $S/\sim_{\mathcal{T}}$

# Partitioning refinement algorithm



$\mathcal{B}_0 := \mathcal{B}_{AP}$

$i := 0$

$\mathcal{B}_0 := \mathcal{B}_{AP}$ $\longleftarrow$ identifies states with the same labeling

$i := 0$

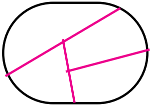
# Partitioning refinement algorithm

$\mathcal{B}_0 := \mathcal{B}_{AP}$ $\leftarrow$ identifies states with the same labeling

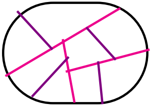
$i := 0$

REPEAT $\mathcal{B}_{i+1} := \mathit{Refine}(\mathcal{B}_i)$

$\mathcal{B}_i$



$\mathcal{B}_{i+1}$



# Partitioning refinement algorithm



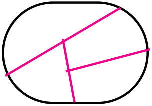
$\mathcal{B}_0 := \mathcal{B}_{AP}$ ⟵ identifies states with the same labeling

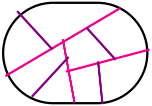
$i := 0$

REPEAT $\mathcal{B}_{i+1} := Refine(\mathcal{B}_i)$

$i := i+1$

UNTIL $\mathcal{B}_i = \mathcal{B}_{i-1}$ ⟵ no more refinement possible

$\mathcal{B}_i$ 

$\mathcal{B}_{i+1}$ 

# Partitioning refinement algorithm

$\mathcal{B}_0 := \mathcal{B}_{AP}$ $\longleftarrow$ identifies states with the same labeling

$i := 0$

REPEAT $\mathcal{B}_{i+1} := \mathit{Refine}(\mathcal{B}_i)$

$i := i+1$

UNTIL $\mathcal{B}_i = \mathcal{B}_{i-1}$ $\longleftarrow$ no more refinement possible
hence: $\mathcal{B}_i = S/{\sim_\mathcal{T}}$

$\mathcal{B}_i$ $\mathcal{B}_{i+1}$

# Partitioning refinement algorithm



$\mathcal{B}_0 := \mathcal{B}_{AP}$ ← identifies states with the same labeling

$i := 0$

REPEAT $\mathcal{B}_{i+1} := Refine(\mathcal{B}_i)$

$i := i+1$

UNTIL $\mathcal{B}_i = \mathcal{B}_{i-1}$ ← no more refinement possible

hence: $\mathcal{B}_i = S/\sim_{\mathcal{T}}$

return $\mathcal{B}_i$

$\mathcal{B}_0 := \mathcal{B}_{AP}$ ← identifies states with the same labeling

$i := 0$

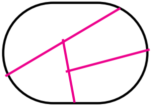
REPEAT $\mathcal{B}_{i+1} := \mathit{Refine}(\mathcal{B}_i)$

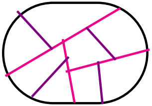
$i := i+1$

UNTIL $\mathcal{B}_i = \mathcal{B}_{i-1}$ ← no more refinement possible

hence: $\mathcal{B}_i = S/\sim_{\mathcal{T}}$

return $\mathcal{B}_i$

$\mathcal{B}_i$ 

$\mathcal{B}_{i+1}$ 

loop invariant:

$\mathcal{B}_i$ is coarser than $S/\sim_{\mathcal{T}}$ and finer than $\mathcal{B}_{AP}$

# Maximal number of iterations?



$\mathcal{B}_0 := \mathcal{B}_{AP}$; $i := 0$

```
REPEAT
```

$\mathcal{B}_{i+1} := \mathit{Refine}(\mathcal{B}_i)$; $i := i+1$

UNTIL no further refinement is possible

$\mathcal{B}_0 := \mathcal{B}_{AP}$; $i := 0$

REPEAT

$\mathcal{B}_{i+1} := \mathit{Refine}(\mathcal{B}_i)$; $i := i+1$

UNTIL no further refinement is possible

Assuming that $\mathcal{B}_i$ is strictly coarser than $\mathcal{B}_{i+1}$ for all $i$, what is the maximal number of refinement steps **?**

# Maximal number of iterations?



$\mathcal{B}_0 := \mathcal{B}_{AP}$; $\;i := 0$

REPEAT

$\quad \mathcal{B}_{i+1} := Refine(\mathcal{B}_i)$; $\;i := i+1$

UNTIL no further refinement is possible

Assuming that $\mathcal{B}_i$ is strictly coarser than $\mathcal{B}_{i+1}$ for all $i$, what is the maximal number of refinement steps ?

**answer:** $|S| - 1$

Note that $|\mathcal{B}_i| \geq i+1$.

# Maximal number of iterations?



$\mathcal{B}_0 := \mathcal{B}_{AP}$; $i := 0$

```
REPEAT
```

$\mathcal{B}_{i+1} := \mathit{Refine}(\mathcal{B}_i)$; $i := i+1$

`UNTIL` no further refinement is possible

Assuming that $\mathcal{B}_i$ is strictly coarser than $\mathcal{B}_{i+1}$ for all $i$, what is the maximal number of refinement steps ?

**answer:** $|S| - 1$

Note that $|\mathcal{B}_i| \geq i+1$.

Hence: if there are $k = |S|-1$ iterations then $\mathcal{B}_k$ consists of singletons

# The initial partition



initial partition $\mathcal{B}_{AP}$:

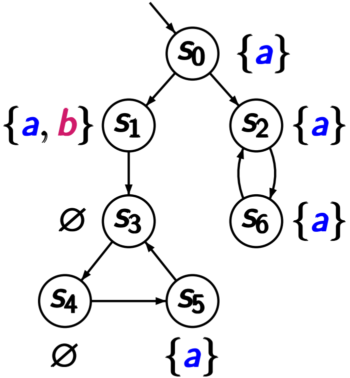
identifies all states $s$, $t$
s.t. $L(s) = L(t)$

# The initial partition



initial partition $\mathcal{B}_{AP}$:

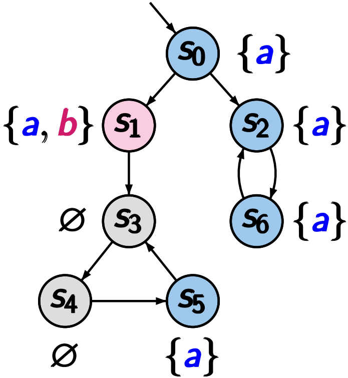
identifies all states $s$, $t$ s.t. $L(s) = L(t)$



# The initial partition

initial partition $\mathcal{B}_{AP}$:

identifies all states $s$, $t$ s.t. $L(s) = L(t)$




$$\mathcal{B}_{AP} = \Big\{ \{s_0, s_2, s_6, s_5\}, \{s_1\}, \{s_3, s_4\} \Big\}$$

# Initial partition



initial partition $\mathcal{B}_{AP}$:

- identifies all states with the same labeling
- agrees with the quotient under the equivalence

$$s \equiv_{AP} t \quad \text{iff} \quad L(s) = L(t)$$

# Initial partition



initial partition $\mathcal{B}_{AP}$:

- identifies all states with the same labeling
- agrees with the quotient under the equivalence

$$s \equiv_{AP} t \quad \text{iff} \quad L(s) = L(t)$$

compute $\mathcal{B}_{AP}$ by an on-the-fly generation of the decision tree for $AP$

# Initial partition



compute $\mathcal{B}_{AP}$ by an on-the-fly generation of the decision tree for $AP = \{a_1, ..., a_k\}$

# Initial partition



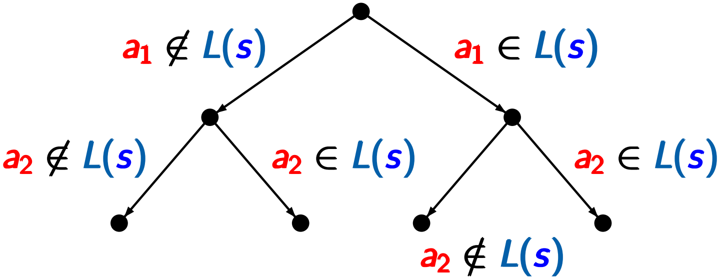
compute $\mathcal{B}_{AP}$ by an on-the-fly generation of the decision tree for $AP = \{a_1, ..., a_k\}$

*inner nodes* at level $i$: decision "$a_i \in L(s)$ ?"

*leaves:* sets of states with the same labeling compute $\mathcal{B}_{AP}$ by an on-the-fly generation of the decision tree for $AP = \{a_1, \ldots, a_k\}$

*inner nodes* at level $i$: decision "$a_i \in L(s)$ ?"

*leaves:* sets of states with the same labeling



## Computing the initial partition



compute $\mathcal{B}_{AP}$ by an on-the-fly generation of the decision tree for $AP = \{a_1, \ldots, a_k\}$

# Computing the initial partition



compute $\mathcal{B}_{AP}$ by an on-the-fly generation of the decision tree for $AP = \{a_1, \dots, a_k\}$

initally: each leaf represents the empty state-set

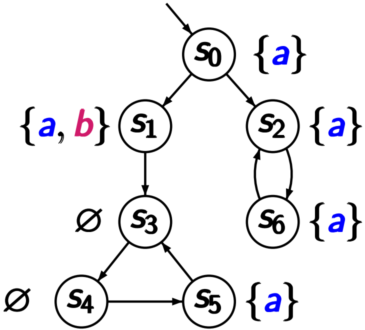
for each state $s$:

traverse the decision tree from the root to a leaf $v$

insert $s$ in the set for $v$

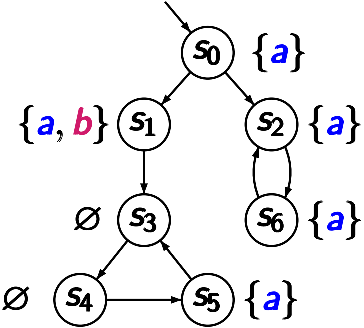
# Example: initial partition





# Example: initial partition



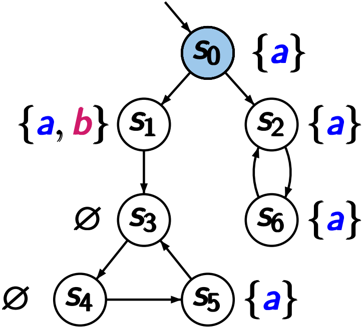


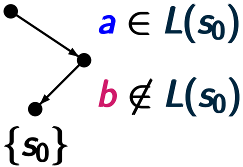

decision tree for
$AP = \{a, b\}$

1. level: $a \in L(s)$ ?
2. level: $b \in L(s)$ ?

# Example: initial partition





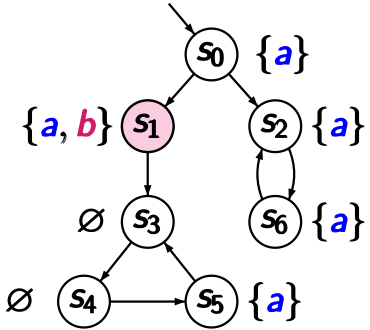




decision tree for
$AP = \{a, b\}$

1. level: $a \in L(s)$ ?
2. level: $b \in L(s)$ ?

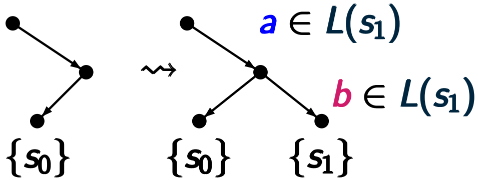
# Example: initial partition





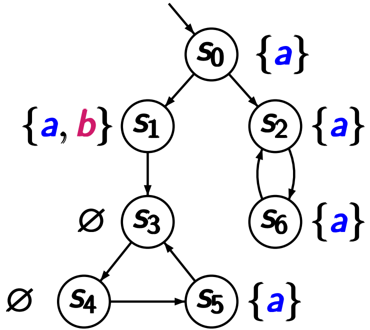

decision tree for
$AP = \{a, b\}$

1. level: $a \in L(s)$ ?
2. level: $b \in L(s)$ ?



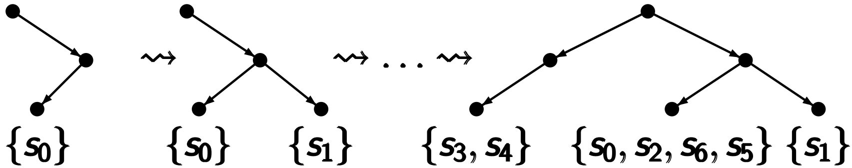
# Example: initial partition





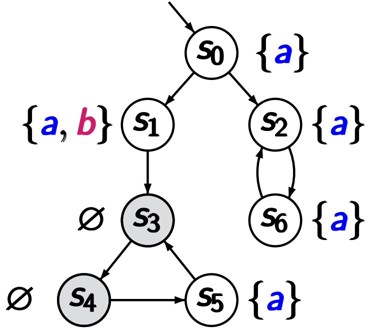

decision tree for
$AP = \{a, b\}$

1. level: $a \in L(s)$ ?
2. level: $b \in L(s)$ ?



# Example: initial partition

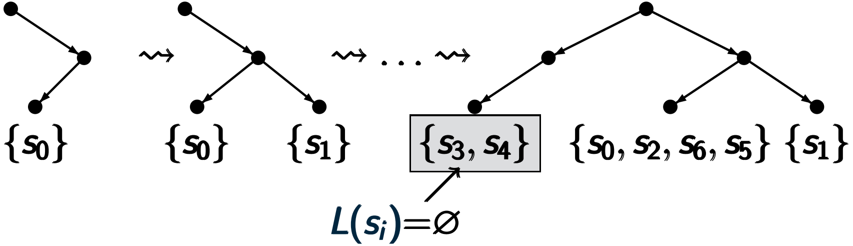





decision tree for

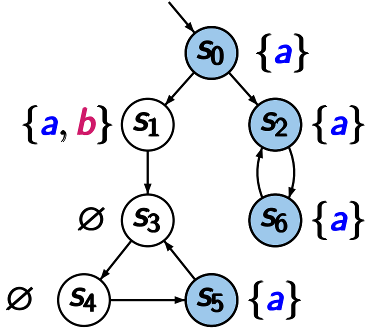
$AP = \{a, b\}$

1. level: $a \in L(s)$ ?
2. level: $b \in L(s)$ ?



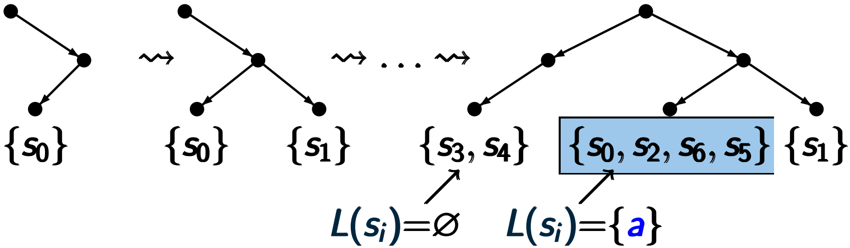
# Example: initial partition





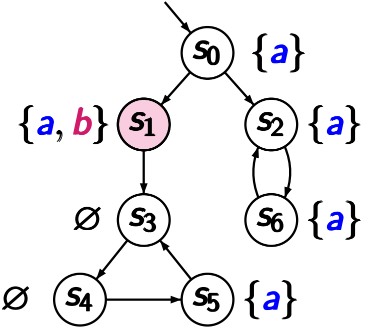

decision tree for
$AP = \{a, b\}$

1. level: $a \in L(s)$ ?
2. level: $b \in L(s)$ ?



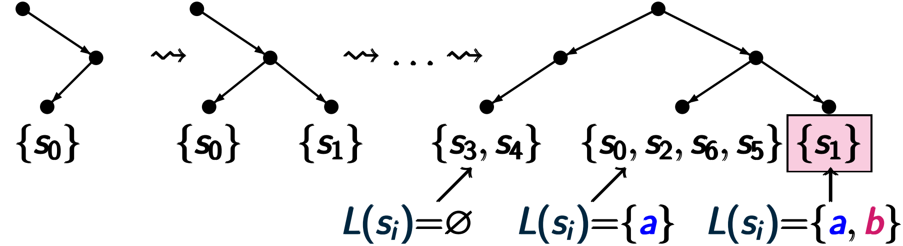
# Example: initial partition






decision tree for
$AP = \{a, b\}$

1. level: $a \in L(s)$ ?
2. level: $b \in L(s)$ ?



# Computation of the initial partition

```
generate the root node v_0 of the decision tree
FOR ALL states s DO
   v := v_0
   FOR i = 1, ..., k OD
      IF a_i ∈ L(s)
         THEN   v := find_or_add(right son of v)
         ELSE   v := find_or_add(left son of v)
      FI
   OD

OD
```

suppose
$AP = \{a_1, \ldots, a_k\}$

# Computation of the initial partition



generate the root node $v_0$ of the decision tree

```
FOR ALL states s DO
   v := v_0
   FOR i = 1, ..., k OD
      IF a_i ∈ L(s)
         THEN  v := find_or_add(right son of v)
         ELSE  v := find_or_add(left son of v)
      FI
   OD  ⟵ v is a leaf of depth k

OD
```

suppose
$AP = \{a_1, \ldots, a_k\}$

```
generate the root node v0 of the decision tree
FOR ALL states s DO
  v := v0
  FOR i = 1,...,k OD
    IF ai ∈ L(s)
       THEN  v := find_or_add(right son of v)
       ELSE  v := find_or_add(left son of v)
    FI
  OD  ⟵  v is a leaf of depth k
  add s into the state-set of v
OD
```

suppose $AP = \{a_1, \ldots, a_k\}$

```
generate the root node v0 of the decision tree
FOR ALL  states s DO
  v := v0
  FOR  i = 1,...,k OD
    IF  a_i ∈ L(s)
        THEN    v := find_or_add(right son of v)
        ELSE    v := find_or_add(left son of v)
    FI
  OD  ⟵  v is a leaf of depth k
  add s into the state-set of v
OD
```

suppose
$AP = \{a_1, \ldots, a_k\}$

The state-sets of the leaves are the blocks in $\mathcal{B}_{AP}$.

# Computation of the initial partition



```
generate the root node v_0 of the decision tree
FOR ALL  states s DO
   v := v_0
   FOR  i = 1, ..., k OD
      IF  a_i ∈ L(s)
           THEN    v := find_or_add(right son of v)
           ELSE    v := find_or_add(left son of v)
      FI
   OD  ⟵ v is a leaf of depth k
   add s into the state-set of v
OD
```

**complexity:**
$\mathcal{O}(|S| \cdot |AP|)$

The state-sets of the leaves are the blocks in $\mathcal{B}_{AP}$.

# Partitioning refinement (schema)



$\mathcal{B} := \mathcal{B}_{AP}$

`WHILE` refinements are possible `DO`

$\qquad \mathcal{B} := \mathit{Refine}(\mathcal{B})$

`OD`

return $\mathcal{B}$

# Partitioning refinement (schema)



$\mathcal{B} := \mathcal{B}_{AP} \longleftarrow$ complexity: $\mathcal{O}(|S| \cdot |AP|)$

```
WHILE refinements are possible DO

        B := Refine(B)
OD

return B
```

# Partitioning refinement (schema)



$\mathcal{B} := \mathcal{B}_{AP}$ ⟵ complexity: $\mathcal{O}(|S| \cdot |AP|)$

```
WHILE refinements are possible DO
```

$$\mathcal{B} := \mathit{Refine}(\mathcal{B})$$

```
OD
```

return $\mathcal{B}$ ⟵ $\boxed{\mathcal{B} = S/{\sim_{\mathcal{T}}}}$

# Partitioning refinement (schema)

$\mathcal{B} := \mathcal{B}_{AP}$

`WHILE` refinements are possible `DO`

$$\boxed{\mathcal{B} := \mathit{Refine}(\mathcal{B})}$$

`OD`

return $\mathcal{B}$

# Partitioning refinement (schema)



$\mathcal{B} := \mathcal{B}_{AP}$

WHILE refinements are possible DO

$\boxed{\mathcal{B} := \mathit{Refine}(\mathcal{B})}$

OD

return $\mathcal{B}$

refinement: stabilization for some superblock $C$ of $\mathcal{B}$:

# Partitioning refinement (schema)



$\mathcal{B} := \mathcal{B}_{AP}$

`WHILE` refinements are possible `DO`

$\boxed{\mathcal{B} := \mathit{Refine}(\mathcal{B})}$

`OD`

return $\mathcal{B}$

refinement: stabilization for some superblock $C$ of $\mathcal{B}$:

split each block $B \in \mathcal{B}$ into two blocks:

$B \cap \mathit{Pre}(C)$ and $B \setminus \mathit{Pre}(C)$

# Partitioning refinement (schema)

$\mathcal{B} := \mathcal{B}_{AP}$

WHILE refinements are possible DO

$\boxed{\mathcal{B} := \mathit{Refine}(\mathcal{B}, B) \text{ for some splitter } C}$

OD

return $\mathcal{B}$

refinement: stabilization for some superblock $C$ of $\mathcal{B}$:

split each block $B \in \mathcal{B}$ into two blocks:

$B \cap \mathit{Pre}(C)$ and $B \setminus \mathit{Pre}(C)$

$\mathcal{B} := \mathcal{B}_{AP}$

`WHILE` refinements are possible `DO`

$\boxed{\mathcal{B} := \mathit{Refine}(\mathcal{B}, B) \text{ for some splitter } C}$

`OD`

return $\mathcal{B}$

refinement: stabilization for some superblock $C$ of $\mathcal{B}$:

split each block $B \in \mathcal{B}$ into two blocks:

$B \cap \mathit{Pre}(C)$ and $B \setminus \mathit{Pre}(C)$

# Partitioning splitter algorithm



$\mathcal{B} := \mathcal{B}_{AP}$

WHILE refinements are possible DO

choose some superblock $C$ of $\mathcal{B}$;

$\mathcal{B} := \mathit{Refine}(\mathcal{B}, B)$

OD

return $\mathcal{B}$

$\mathcal{B} := \mathcal{B}_{AP}$

`WHILE` refinements are possible `DO`

  choose some superblock $C$ of $\mathcal{B}$;

  $\mathcal{B} := \mathit{Refine}(\mathcal{B}, B) = \bigcup_{B \in \mathcal{B}} \mathit{Refine}(B, C)$

`OD`

return $\mathcal{B}$

# Partitioning splitter algorithm



$\mathcal{B} := \mathcal{B}_{AP}$

WHILE refinements are possible DO

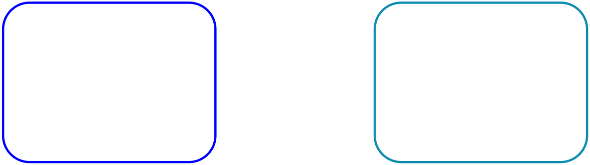
  choose some superblock $C$ of $\mathcal{B}$;

  $\mathcal{B} := Refine(\mathcal{B}, B) = \bigcup\limits_{B \in \mathcal{B}} Refine(B, C)$

OD

return $\mathcal{B}$

$Refine(B, C)$



block $B$ superblock $C$

$\mathcal{B} := \mathcal{B}_{AP}$

WHILE refinements are possible DO

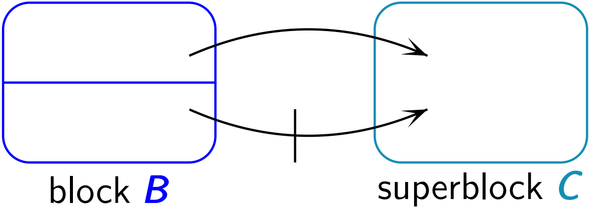
  choose some superblock $C$ of $\mathcal{B}$;

  $\mathcal{B} := Refine(\mathcal{B}, B) = \bigcup\limits_{B \in \mathcal{B}} Refine(B, C)$

OD

return $\mathcal{B}$

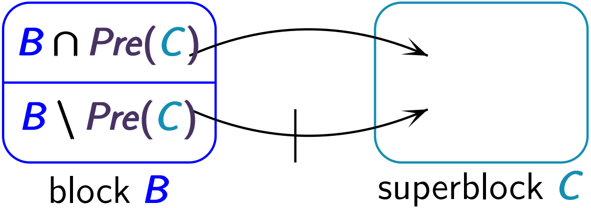
$Refine(B, C)$



# Partitioning splitter algorithm

$\mathcal{B} := \mathcal{B}_{AP}$
WHILE refinements are possible DO
  choose some superblock $C$ of $\mathcal{B}$;
  $\mathcal{B} := Refine(\mathcal{B}, B) = \bigcup_{B \in \mathcal{B}} Refine(B, C)$
OD
return $\mathcal{B}$

$$Refine(B, C) = \{B \cap Pre(C), B \setminus Pre(C)\}$$



$\mathcal{B} := \mathcal{B}_{AP}$

WHILE refinements are possible DO

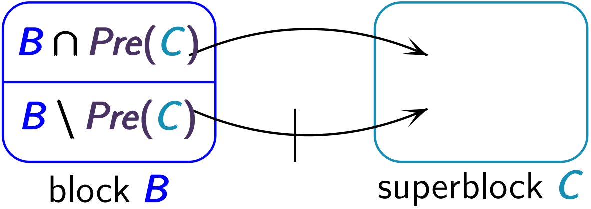
  choose some superblock $C$ of $\mathcal{B}$;

  $\mathcal{B} := Refine(\mathcal{B}, B) = \bigcup_{B \in \mathcal{B}} Refine(B, C)$

OD

return $\mathcal{B}$

$$Refine(B, C) = \{B \cap Pre(C), B \setminus Pre(C)\} \setminus \{\varnothing\}$$



## The refinement operator



Let $\mathcal{B}$ be a partition for $S$ and $C$ a superblock of $\mathcal{B}$.

$$Refine(\mathcal{B}, C) = \bigcup_{B \in \mathcal{B}} Refine(B, C)$$

where $Refine(B, C) = \{B \cap Pre(C), B \setminus Pre(C)\} \setminus \{\varnothing\}$

## The refinement operator

Let $\mathcal{B}$ be a partition for $S$ and $C$ a superblock of $\mathcal{B}$.

$$Refine(\mathcal{B}, C) = \bigcup_{B \in \mathcal{B}} Refine(B, C)$$

where $Refine(B, C) = \{B \cap Pre(C), B \setminus Pre(C)\} \setminus \{\varnothing\}$

If $\mathcal{B}$ is finer than $\mathcal{B}_{AP}$ and coarser than $S/\!\sim_{\mathcal{T}}$ then:

Let $\mathcal{B}$ be a partition for $S$ and $C$ a superblock of $\mathcal{B}$.

$$Refine(\mathcal{B}, C) = \bigcup_{B \in \mathcal{B}} Refine(B, C)$$

where $Refine(B, C) = \{ B \cap Pre(C), B \setminus Pre(C) \} \setminus \{\varnothing\}$

If $\mathcal{B}$ is finer than $\mathcal{B}_{AP}$ and coarser than $S/{\sim_{\mathcal{T}}}$ then:

(a) $Refine(\mathcal{B}, C)$ is finer than $\mathcal{B}$

# The refinement operator



Let $\mathcal{B}$ be a partition for $S$ and $C$ a superblock of $\mathcal{B}$.

$$Refine(\mathcal{B}, C) = \bigcup_{B \in \mathcal{B}} Refine(B, C)$$

where $Refine(B, C) = \left\{ B \cap Pre(C), B \setminus Pre(C) \right\} \setminus \{\varnothing\}$

If $\mathcal{B}$ is finer than $\mathcal{B}_{AP}$ and coarser than $S/\!\sim_{\mathcal{T}}$ then:

(a) $Refine(\mathcal{B}, C)$ is finer than $\mathcal{B}$ and $\mathcal{B}_{AP}$

# The refinement operator



Let $\mathcal{B}$ be a partition for $S$ and $C$ a superblock of $\mathcal{B}$.

$$Refine(\mathcal{B}, C) = \bigcup_{B \in \mathcal{B}} Refine(B, C)$$

where $Refine(B, C) = \{B \cap Pre(C), B \setminus Pre(C)\} \setminus \{\varnothing\}$

If $\mathcal{B}$ is finer than $\mathcal{B}_{AP}$ and coarser than $S/\!\sim_{\mathcal{T}}$ then:

(a) $Refine(\mathcal{B}, C)$ is finer than $\mathcal{B}$ and $\mathcal{B}_{AP}$

(b) $Refine(\mathcal{B}, C)$ is coarser than $S/\!\sim_{\mathcal{T}}$

## The refinement operator



Let $\mathcal{B}$ be a partition for $S$ and $C$ a superblock of $\mathcal{B}$.

$$Refine(\mathcal{B}, C) = \bigcup_{B \in \mathcal{B}} Refine(B, C)$$

where $Refine(B, C) = \left\{ B \cap Pre(C), B \setminus Pre(C) \right\} \setminus \{\varnothing\}$

If $\mathcal{B}$ is finer than $\mathcal{B}_{AP}$ and coarser than $S/{\sim_{\mathcal{T}}}$ then:

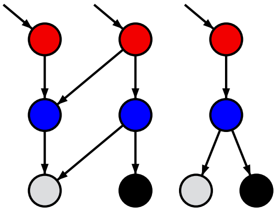
(a) $Refine(\mathcal{B}, C)$ is finer than $\mathcal{B}$ and $\mathcal{B}_{AP}$

(b) $Refine(\mathcal{B}, C)$ is coarser than $S/{\sim_{\mathcal{T}}}$

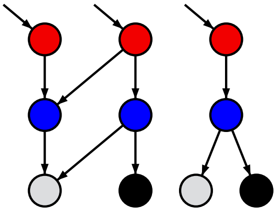
(c) $Refine(\mathcal{B}, C) = \mathcal{B}$ for all $C \in \mathcal{B}$ iff $\mathcal{B} = S/{\sim_{\mathcal{T}}}$

# Example: partitioning splitter algorithm





# Example: partitioning splitter algorithm

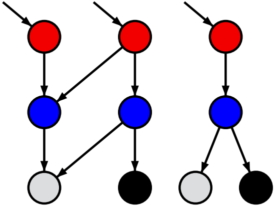


$\Longrightarrow$

refinement
w.r.t. ●

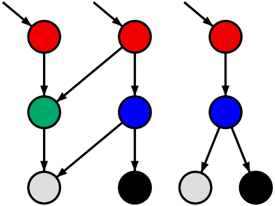
# Example: partitioning splitter algorithm





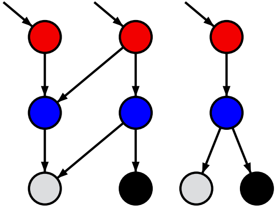
$\Longrightarrow$

refinement
w.r.t.



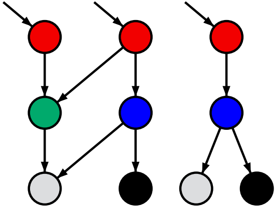
# Example: partitioning splitter algorithm





$\Longrightarrow$

refinement
w.r.t.

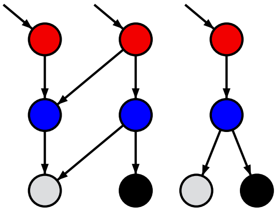


$\Downarrow$

refinement
w.r.t.

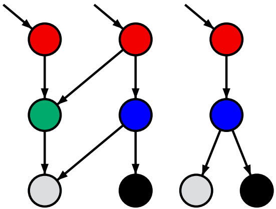
# Example: partitioning splitter algorithm





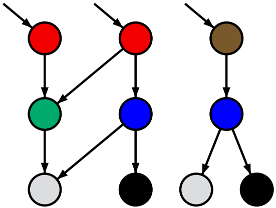
$\Longrightarrow$

refinement
w.r.t.



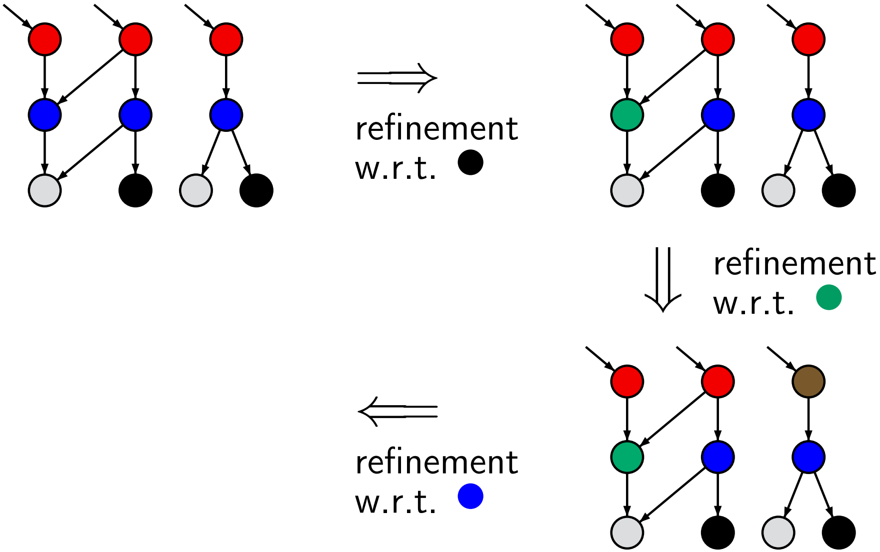
$\Downarrow$

refinement
w.r.t.



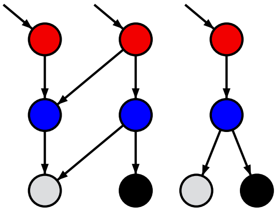
# Example: partitioning splitter algorithm





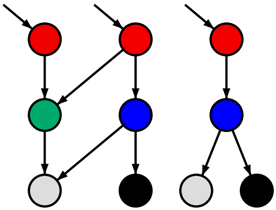
# Example: partitioning splitter algorithm





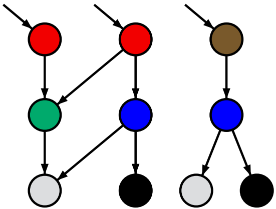
$\Longrightarrow$

refinement
w.r.t.



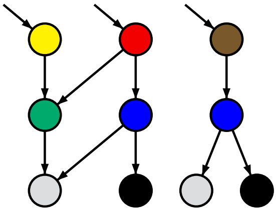
$\Downarrow$

refinement
w.r.t.



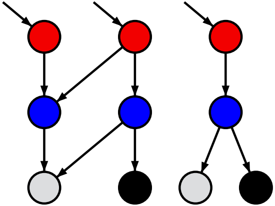
$\Longleftarrow$

refinement
w.r.t.



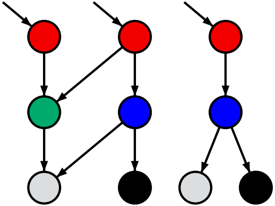
# Example: partitioning splitter algorithm





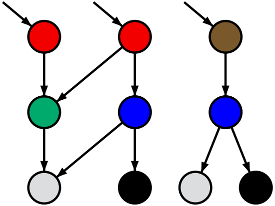
$\Longrightarrow$

refinement
w.r.t.



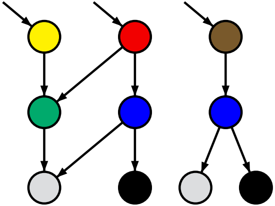
$\Downarrow$

refinement
w.r.t.



$\Longleftarrow$

refinement
w.r.t.



**7** bisimulation equivalence classes

# The refinement operator



given a partition $\mathcal{B}$ and a superblock $C$ of $\mathcal{B}$,
how to compute

$$\mathit{Refine}(\mathcal{B}, C)$$

efficiently **?**

# The refinement operator



given a partition $\mathcal{B}$ and a superblock $C$ of $\mathcal{B}$,
how to compute

$$Refine(\mathcal{B}, C) = \bigcup_{B \in \mathcal{B}} Refine(B, C)$$

efficiently **?**

# The refinement operator

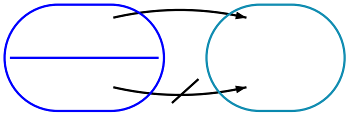
given a partition $\mathcal{B}$ and a superblock $C$ of $\mathcal{B}$, how to compute

$$Refine(\mathcal{B}, C) = \bigcup_{B \in \mathcal{B}} Refine(B, C)$$

efficiently **?**

where for all blocks $B \in \mathcal{B}$:

$$Refine(B, C) = \{\, B \cap Pre(C),\ B \setminus Pre(C) \,\} \setminus \{\varnothing\}$$



block $B$ superblock $C$

# Refinement operator $Refine(\mathcal{B}, C)$

# Refinement operator $Refine(\mathcal{B}, C)$

```
FOR ALL s' ∈ C DO



OD
```

# Refinement operator $Refine(\mathcal{B}, C)$



```
FOR ALL s' ∈ C DO
   FOR ALL s ∈ Pre(s') DO



   OD
OD
```

# Refinement operator $Refine(\mathcal{B}, C)$



```
FOR ALL s' ∈ C DO
   FOR ALL s ∈ Pre(s') DO
         "move" state s from block [s]_B = B
                to the new block B ∩ Pre(C)
   OD
OD
```

## Refinement operator $Refine(\mathcal{B}, C)$

```
FOR ALL s′ ∈ C DO
   FOR ALL s ∈ Pre(s′) DO
         "move" state s from block [s]_B = B
                  to the new block B ∩ Pre(C)
   OD
OD
```

... states left in block $B \in \mathcal{B}$ belong to the new block $B \setminus Pre(C)$

```
FOR ALL s' ∈ C DO
   FOR ALL s ∈ Pre(s') DO
        "move" state s from block [s]_B = B
              to the new block B ∩ Pre(C)
   OD
OD
```

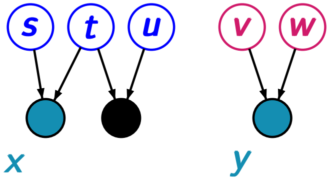
... states left in block $B \in \mathcal{B}$ belong to the new block $B \setminus \mathit{Pre}(C)$

time complexity:

$$\mathcal{O}\Big( \sum_{s' \in C} |\mathit{Pre}(s')| + |C| \Big)$$

# Example: refinement operator

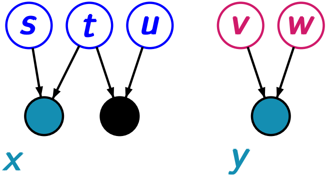





partition $\mathcal{B}$

# Example: refinement operator

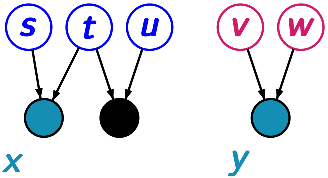





superblock $C = \{x, y\}$

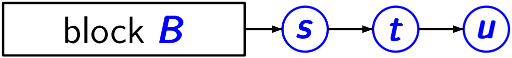
partition $\mathcal{B} \rightsquigarrow \mathit{Refine}(\mathcal{B}, C)$

# Example: refinement operator




superblock $C = \{x, y\}$

partition $\mathcal{B} \rightsquigarrow \mathit{Refine}(\mathcal{B}, C)$

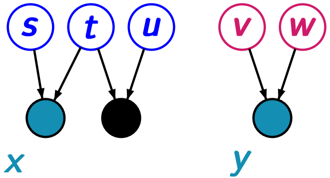




...

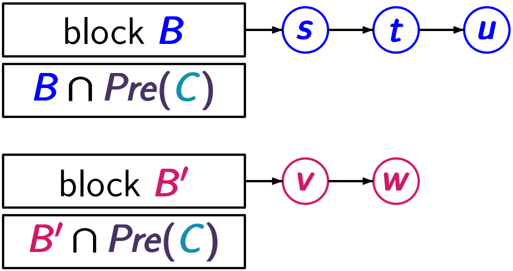
# Example: refinement operator





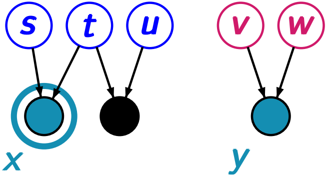

superblock $C = \{x, y\}$

partition $\mathcal{B} \rightsquigarrow \mathit{Refine}(\mathcal{B}, C)$




...

# Example: refinement operator




superblock $C = \{x, y\}$

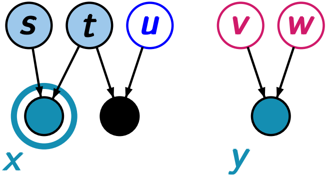
partition $\mathcal{B} \rightsquigarrow Refine(\mathcal{B}, C)$

| block $B$ | $\rightarrow s \rightarrow t \rightarrow u$ |
|---|---|
| $B \cap Pre(C)$ | |

| block $B'$ | $\rightarrow v \rightarrow w$ |
|---|---|
| $B' \cap Pre(C)$ | |

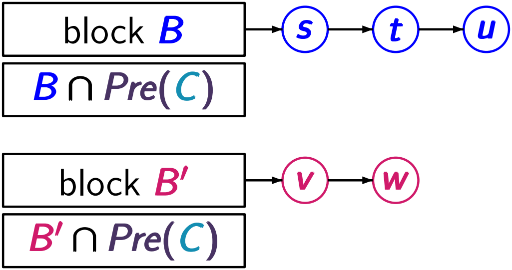
...

# Example: refinement operator




superblock $C = \{x, y\}$

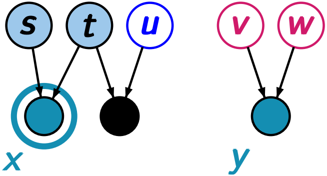
partition $\mathcal{B} \rightsquigarrow \mathit{Refine}(\mathcal{B}, C)$




...

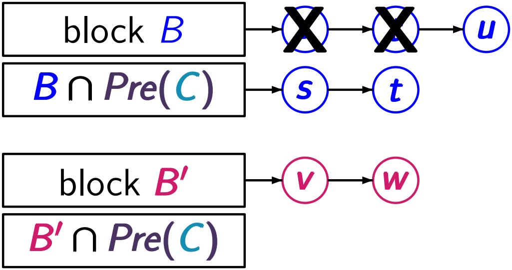
# Example: refinement operator






superblock $C = \{x, y\}$

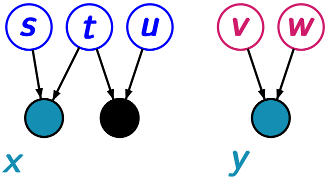
partition $\mathcal{B} \rightsquigarrow \mathit{Refine}(\mathcal{B}, C)$




...

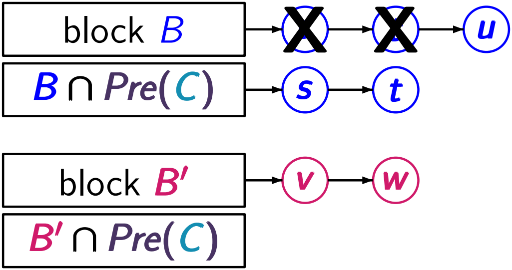
# Example: refinement operator






superblock $C = \{x, y\}$

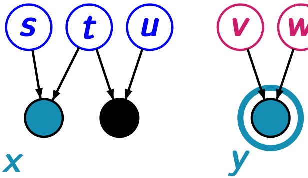
partition $\mathcal{B} \rightsquigarrow \mathit{Refine}(\mathcal{B}, C)$




$\longleftarrow$ new block $B \setminus Pre(C)$

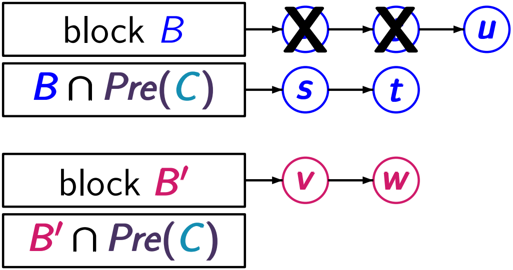
...

# Example: refinement operator




superblock $C = \{x, y\}$

partition $\mathcal{B} \rightsquigarrow \mathit{Refine}(\mathcal{B}, C)$

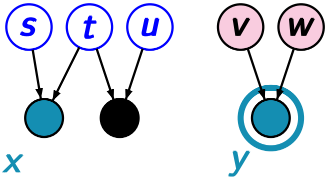



$\leftarrow$ new block $B \setminus \mathit{Pre}(C)$

...

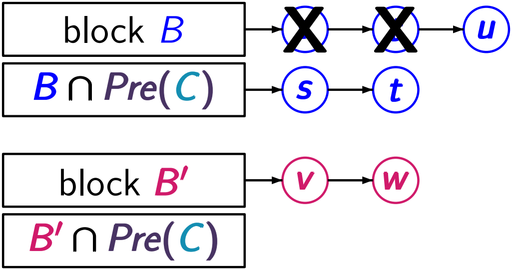
# Example: refinement operator






superblock $C = \{x, y\}$

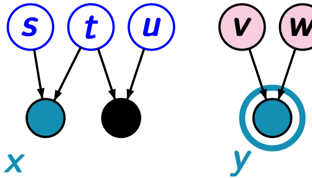
partition $\mathcal{B} \rightsquigarrow \mathit{Refine}(\mathcal{B}, C)$




$\leftarrow$ new block $B \setminus \mathit{Pre}(C)$

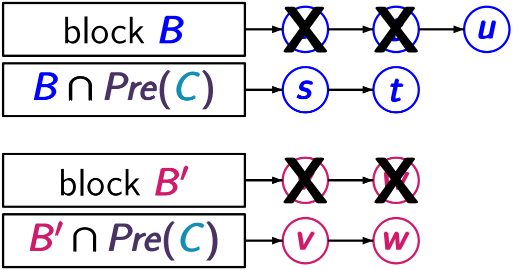
...

# Example: refinement operator




superblock $C = \{x, y\}$

partition $\mathcal{B} \rightsquigarrow \mathit{Refine}(\mathcal{B}, C)$

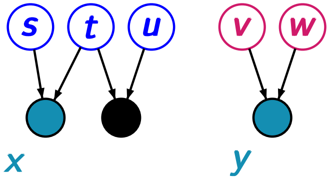



$\leftarrow$ new block $B \setminus \mathit{Pre}(C)$

$\ldots$

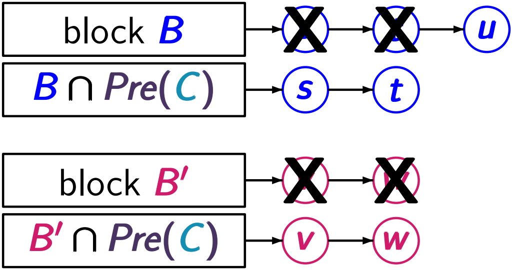
# Example: refinement operator






superblock $C = \{x, y\}$

partition $\mathcal{B} \rightsquigarrow \mathit{Refine}(\mathcal{B}, C)$

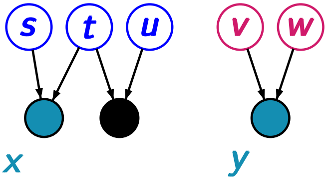



$\leftarrow$ new block $B \setminus \mathit{Pre}(C)$

$\leftarrow$ $B' \setminus \mathit{Pre}(C) = \varnothing$

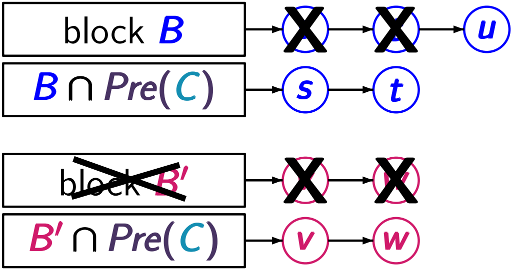
...

# Example: refinement operator




superblock $C = \{x, y\}$

partition $\mathcal{B} \rightsquigarrow \mathit{Refine}(\mathcal{B}, C)$

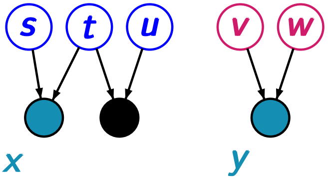



$\leftarrow$ new block $B \setminus Pre(C)$

$\leftarrow B' \setminus Pre(C) = \varnothing$

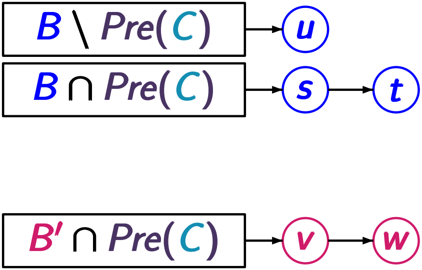
...

# Example: refinement operator




superblock $C = \{x, y\}$

$Refine(\mathcal{B}, C)$




...

# Partitioning splitter algorithm



$\mathcal{B} := \mathcal{B}_{AP}$

`WHILE` there is a splitter $C$ for $\mathcal{B}$ `DO`

  select such a splitter $C$;
  $\mathcal{B} := \mathit{Refine}(\mathcal{B}, C)$

`OD`

return $\mathcal{B}$

# Partitioning splitter algorithm



$\mathcal{B} := \mathcal{B}_{AP}$ $\longleftarrow$ $\boxed{\text{time complexity: } \mathcal{O}(|S| \cdot |AP|)}$

WHILE there is a splitter $C$ for $\mathcal{B}$ DO

    select such a splitter $C$;
    $\mathcal{B} := Refine(\mathcal{B}, C)$

OD

return $\mathcal{B}$

# Partitioning splitter algorithm



$\mathcal{B} := \mathcal{B}_{AP}$ $\longleftarrow$ $\boxed{\text{time complexity: } \mathcal{O}(|S| \cdot |AP|)}$

WHILE there is a splitter $C$ for $\mathcal{B}$ DO

select such a splitter $C$;

$\mathcal{B} := \mathit{Refine}(\mathcal{B}, C)$

OD

return $\mathcal{B}$

each state $s' \in C$ causes the costs $\mathcal{O}(|\mathit{Pre}(s')| + 1)$

# Partitioning splitter algorithm

$\mathcal{B} := \mathcal{B}_{AP}$ $\longleftarrow$ $\boxed{\text{time complexity: } \mathcal{O}(|S| \cdot |AP|)}$

WHILE there is a splitter $C$ for $\mathcal{B}$ DO

  select such a splitter $C$;

  $\mathcal{B} := \mathit{Refine}(\mathcal{B}, C)$

OD

return $\mathcal{B}$

each state $s' \in C$ causes the costs $\mathcal{O}(|\mathit{Pre}(s')| + 1)$

time complexity:

$$\mathcal{O}\Big( \sum_{C} \big( \sum_{s' \in C} |\mathit{Pre}(s')| + |C| \big) \; + \; |S| \cdot |AP| \Big)$$

# Partitioning splitter algorithm



$\mathcal{B} := \mathcal{B}_{AP}$ $\longleftarrow$ $\boxed{\text{time complexity: } \mathcal{O}(|S| \cdot |AP|)}$

`WHILE` there is a splitter $C$ for $\mathcal{B}$ `DO`

select such a splitter $C$;

$\mathcal{B} := Refine(\mathcal{B}, C)$

`OD`

return $\mathcal{B}$

each state $s' \in C$ causes the costs $\mathcal{O}(|Pre(s')| + 1)$

time complexity:

$$\mathcal{O}\Big( \sum_{C} \big( \sum_{s' \in C} |Pre(s')| + |C| \big) \ + \ |S| \cdot |AP| \Big)$$

$+$ cost for splitter search and management

# Partitioning splitter algorithms



**2** instances of the partitioning splitter algorithm
that differ in the choice and management of splitters

# Partitioning splitter algorithms



**2** instances of the partitioning splitter algorithm
that differ in the choice and management of splitters

- *Kanellakis-Smolka algorithm:*

  refinement according to all blocks of the partition of the previous iteration

## Partitioning splitter algorithms

2 instances of the partitioning splitter algorithm that differ in the choice and management of splitters

- *Kanellakis-Smolka algorithm:*

  refinement according to all blocks of the partition of the previous iteration

- *Paige-Tarjan-algorithm:*

  simultaneous refinement according to 2 superblocks

# Kanellakis-Smolka algorithm

$\mathcal{B} := \mathcal{B}_{AP}$; $\mathcal{B}_{\text{old}} := \{S\}$

REPEAT

    $\mathcal{B}_{\text{old}} := \mathcal{B}$;

    FOR ALL $C \in \mathcal{B}_{\text{old}}$ DO $\mathcal{B} := \mathit{Refine}(\mathcal{B}, C)$ OD

UNTIL $\mathcal{B} = \mathcal{B}_{\text{old}}$

return $\mathcal{B}$

# Kanellakis-Smolka algorithm



$\mathcal{B} := \mathcal{B}_{AP}$; $\mathcal{B}_{\mathsf{old}} := \{S\}$ $\longleftarrow$ $\boxed{\text{cost: } \mathcal{O}(|S| \cdot |AP|)}$

```
REPEAT
    B_old := B;
    FOR ALL C ∈ B_old DO B := Refine(B, C) OD
UNTIL B = B_old
return B
```

$\mathcal{B} := \mathcal{B}_{AP}$; $\mathcal{B}_{\text{old}} := \{S\}$ $\longleftarrow$ $\boxed{\text{cost: } \mathcal{O}(|S|\cdot|AP|)}$

```
REPEAT
    B_old := B;
    FOR ALL C ∈ B_old DO B := Refine(B, C) OD
UNTIL B = B_old
return B
```

- maximal $|S|$ iterations

# Kanellakis-Smolka algorithm



$\mathcal{B} := \mathcal{B}_{AP}$; $\mathcal{B}_{\text{old}} := \{S\}$ $\longleftarrow$ $\boxed{\text{cost: } \mathcal{O}(|S| \cdot |AP|)}$

REPEAT

  $\mathcal{B}_{\text{old}} := \mathcal{B}$;

  FOR ALL $C \in \mathcal{B}_{\text{old}}$ DO $\mathcal{B} := \mathit{Refine}(\mathcal{B}, C)$ OD

UNTIL $\mathcal{B} = \mathcal{B}_{\text{old}}$

return $\mathcal{B}$

- maximal $|S|$ iterations
- per iteration: each state $s' \in C$ causes the costs

$$\mathcal{O}(|\mathit{Pre}(s')| + 1)$$

$\mathcal{B} := \mathcal{B}_{AP}$; $\mathcal{B}_{old} := \{S\}$ $\longleftarrow$ $\boxed{\text{cost: } \mathcal{O}(|S| \cdot |AP|)}$

```
REPEAT
    B_old := B;
    FOR ALL C ∈ B_old DO B := Refine(B, C) OD
UNTIL B = B_old
return B
```

$$\begin{array}{l} \texttt{REPEAT} \\ \quad \mathcal{B}_{old} := \mathcal{B}; \\ \quad \texttt{FOR ALL } C \in \mathcal{B}_{old} \texttt{ DO } \mathcal{B} := \mathit{Refine}(\mathcal{B}, C) \texttt{ OD} \\ \texttt{UNTIL } \mathcal{B} = \mathcal{B}_{old} \\ \text{return } \mathcal{B} \end{array}$$

- maximal $|S|$ iterations
- per iteration: each state $s' \in C$ causes the costs
  $$\mathcal{O}(|Pre(s')| + 1)$$
- cost per iteration: $\mathcal{O}(m + |S|)$

## Kanellakis-Smolka algorithm

$\mathcal{B} := \mathcal{B}_{AP}$; $\mathcal{B}_{\text{old}} := \{S\}$ $\longleftarrow$ $\boxed{\text{cost: } \mathcal{O}(|S| \cdot |AP|)}$

REPEAT

  $\mathcal{B}_{\text{old}} := \mathcal{B}$;

  FOR ALL $C \in \mathcal{B}_{\text{old}}$ DO $\mathcal{B} := \mathit{Refine}(\mathcal{B}, C)$ OD

UNTIL $\mathcal{B} = \mathcal{B}_{\text{old}}$

return $\mathcal{B}$

- maximal $|S|$ iterations
- per iteration: each state $s' \in C$ causes the costs
  $$\mathcal{O}(|\mathit{Pre}(s')| + 1)$$
- cost per iteration: $\mathcal{O}(m + |S|)$
  if $m$ = number of edges $= \sum_{s'} |\mathit{Pre}(s')|$

# Kanellakis-Smolka algorithm

```
B := B_AP;  B_old := {S}        <-- cost: O(|S|·|AP|)
REPEAT
    B_old := B;
    FOR ALL  C ∈ B_old DO  B := Refine(B, C) OD
UNTIL  B = B_old
return B
```

$\mathcal{B} := \mathcal{B}_{AP}$; $\mathcal{B}_{\text{old}} := \{S\}$ $\longleftarrow$ $\boxed{\text{cost: } \mathcal{O}(|S|\cdot|AP|)}$

REPEAT

  $\mathcal{B}_{\text{old}} := \mathcal{B}$;

  FOR ALL $C \in \mathcal{B}_{\text{old}}$ DO $\mathcal{B} := \mathit{Refine}(\mathcal{B}, C)$ OD

UNTIL $\mathcal{B} = \mathcal{B}_{\text{old}}$

return $\mathcal{B}$

- maximal $|S|$ iterations
- per iteration: each state $s' \in C$ causes the costs
  $$\mathcal{O}(|\mathit{Pre}(s')| + 1)$$
- cost per iteration: $\mathcal{O}(m + |S|) = \mathcal{O}(m)$
  if $m$ = number of edges $= \sum_{s'} |\mathit{Pre}(s')| \geq |S|$

$\mathcal{B} := \mathcal{B}_{AP}$; $\mathcal{B}_{\mathsf{old}} := \{S\}$

REPEAT

$\quad\mathcal{B}_{\mathsf{old}} := \mathcal{B}$;

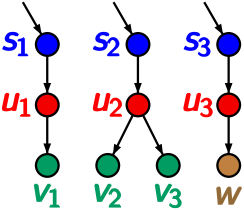
$\quad$FOR ALL $C \in \mathcal{B}_{\mathsf{old}}$ DO $\mathcal{B} := \mathit{Refine}(\mathcal{B}, C)$ OD

UNTIL $\mathcal{B} = \mathcal{B}_{\mathsf{old}}$

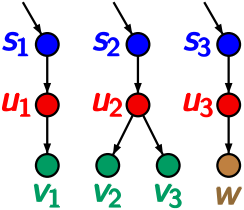
return $\mathcal{B}$

time complexity:
$\mathcal{O}(|S| \cdot m + |S| \cdot |AP|)$

- maximal $|S|$ iterations
- per iteration: each state $s' \in C$ causes the costs
$$\mathcal{O}(|Pre(s')| + 1)$$
- cost per iteration: $\mathcal{O}(m + |S|) = \mathcal{O}(m)$
  if $m$ = number of edges $= \sum_{s'} |Pre(s')| \geq |S|$

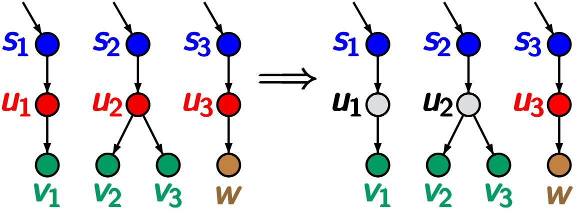

$s_1$
$s_2$
$s_3$
$u_1$
$u_2$
$u_3$
$v_1$
$v_2$
$v_3$
$w$

# Example: Kanellakis-Smolka algorithm



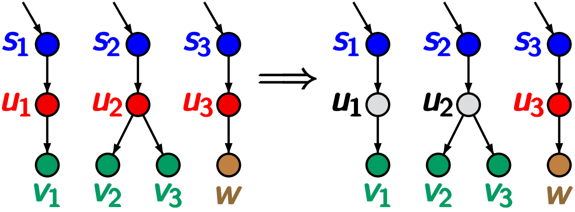

**1. iteration**:

1. refinement w.r.t. $\{v_1, v_2, v_3\}$




**1. iteration**:

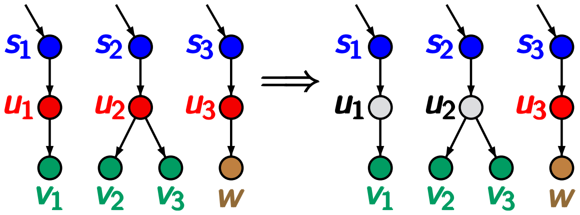
1. refinement w.r.t. $\{v_1, v_2, v_3\}$




**1. iteration**:

1. refinement w.r.t. $\{v_1, v_2, v_3\}$
2. refinement w.r.t. $\{w\}$: no changes

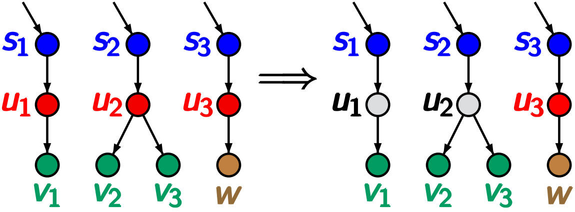
# Example: Kanellakis-Smolka algorithm




**1. iteration**:

1. refinement w.r.t. $\{v_1, v_2, v_3\}$
2. refinement w.r.t. $\{w\}$: no changes
3. refinement w.r.t. $\{s_1, s_2, s_3\}$: no changes

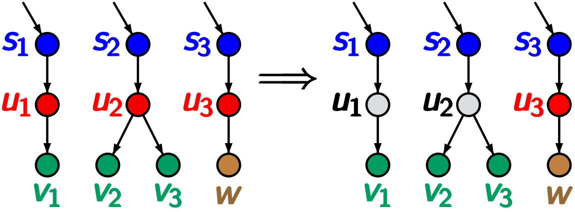
# Example: Kanellakis-Smolka algorithm




**1. iteration**:

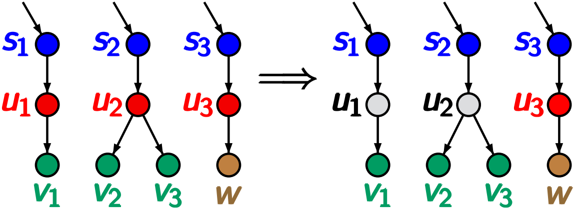
1. refinement w.r.t. $\{v_1, v_2, v_3\}$
2. refinement w.r.t. $\{w\}$: no changes
3. refinement w.r.t. $\{s_1, s_2, s_3\}$: no changes
4. refinement w.r.t. $\{u_1, u_2, u_3\}$: no changes

# Example: Kanellakis-Smolka algorithm




**2. iteration**:

# Example: Kanellakis-Smolka algorithm

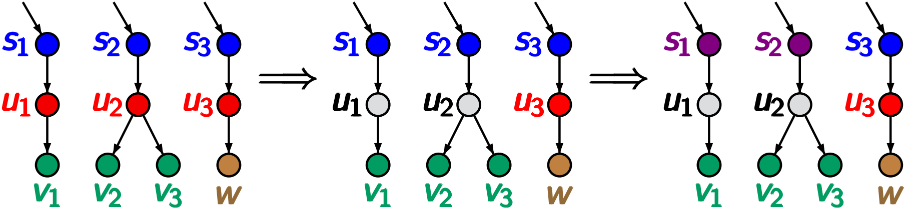



**2. iteration**:

1. refinement w.r.t. $\{u_3\}$

# Example: Kanellakis-Smolka algorithm

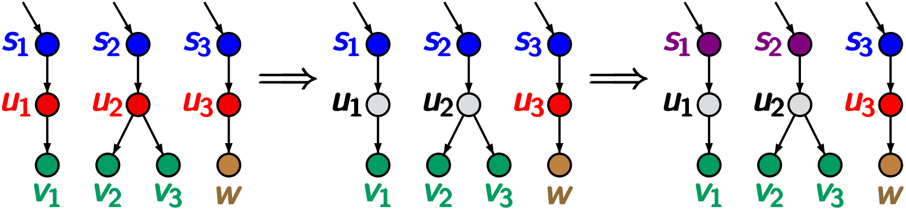





**2. iteration**:

1. refinement w.r.t. $\{u_3\}$

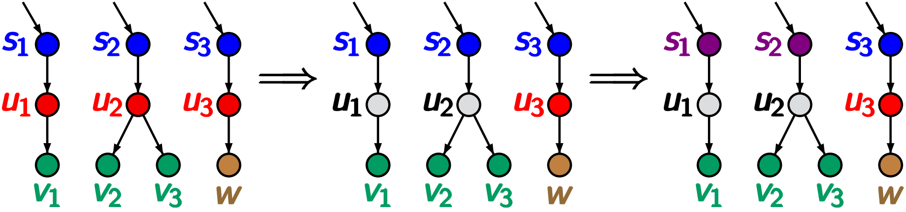
# Example: Kanellakis-Smolka algorithm




**2. iteration**:

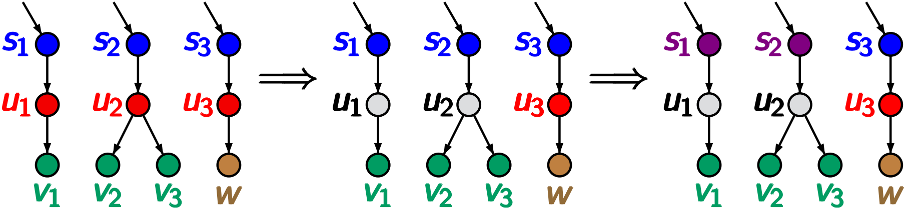
1. refinement w.r.t. $\{u_3\}$
2. refinement w.r.t. other blocks of the first iteration: no changes

# Example: Kanellakis-Smolka algorithm



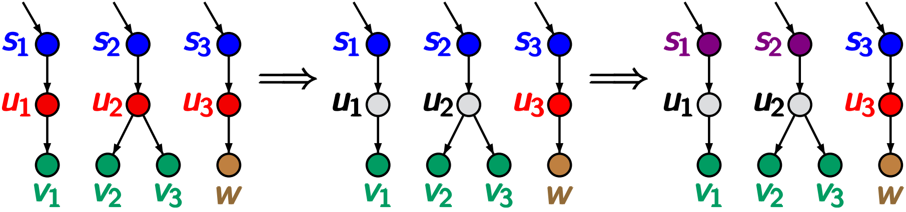

**3. iteration**:

# Example: Kanellakis-Smolka algorithm




**3. iteration**:

refinement w.r.t. all blocks of the second iteration: no changes




**3. iteration**:

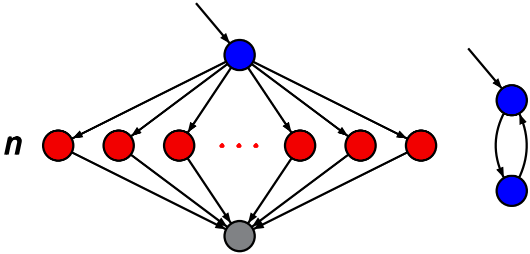
refinement w.r.t. all blocks of the second iteration: no changes

**6** bisimulation equivalence classes:

$\{s_1, s_2\}, \{s_3\}, \{u_1, u_2\}, \{u_3\}, \{v_1, v_2, v_3\}, \{w\}$

# Partitioning splitter algorithm






$AP = \{a, b\}$

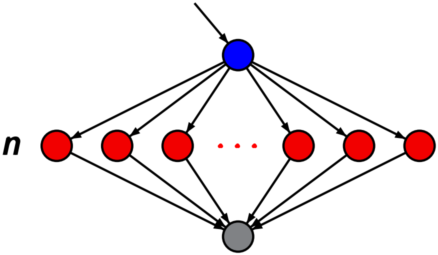
$\bullet \;\widehat{=}\; \{a\}$

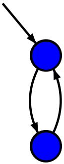
$\bullet \;\widehat{=}\; \{b\}$

$\bullet \;\widehat{=}\; \varnothing$

# Partitioning splitter algorithm





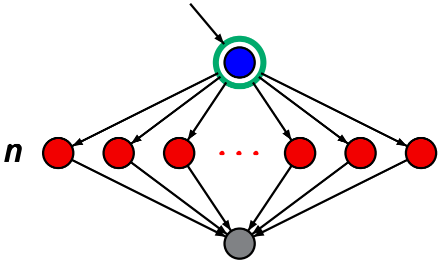
$AP = \{a, b\}$

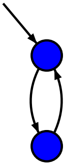
● $\widehat{=}$ $\{a\}$

● $\widehat{=}$ $\{b\}$

● $\widehat{=}$ $\varnothing$

refinement w.r.t. ●:



$AP = \{a, b\}$

● $\widehat{=} \{a\}$

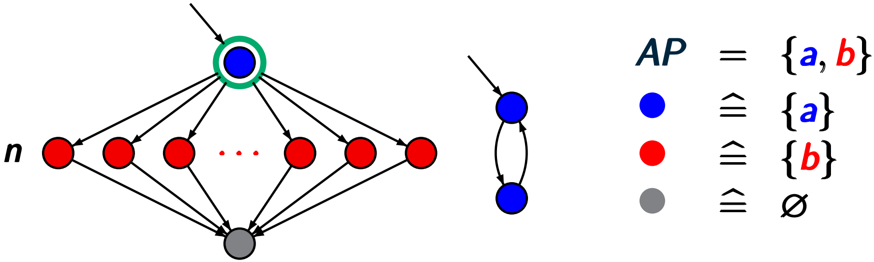
● $\widehat{=} \{b\}$

● $\widehat{=} \varnothing$

refinement w.r.t. ●:

# Partitioning splitter algorithm



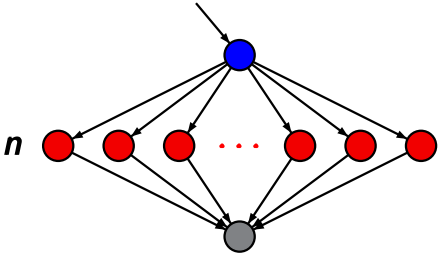


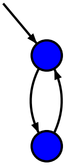

refinement w.r.t. ●: causes the costs

$$\sum_{s'} \left| Pre(s') \right| = n$$

# Partitioning splitter algorithm





$AP = \{a, b\}$

● $\widehat{=}$ $\{a\}$

● $\widehat{=}$ $\{b\}$

● $\widehat{=}$ $\varnothing$

refinement w.r.t. ●: causes the costs

$$\sum_{s'} |Pre(s')| = n$$

alternatively: refinement w.r.t. ●: constant costs

*Kanellakis-Smolka algorithm:*

initially: $\mathcal{B}_{\mathsf{old}} = \mathcal{B} = \mathcal{B}_{AP}$

*Kanellakis-Smolka algorithm:*

initially: $\mathcal{B}_{\mathsf{old}} = \mathcal{B} = \mathcal{B}_{AP}$

iteration: stabilization for each block in $\mathcal{B}_{\mathsf{old}}$

*Kanellakis-Smolka algorithm:*

initially: $\mathcal{B}_{\mathsf{old}} = \mathcal{B} = \mathcal{B}_{AP}$

iteration: stabilization for each block in $\mathcal{B}_{\mathsf{old}}$

loop invariant: $\mathcal{B}$ finer than $\mathcal{B}_{\mathsf{old}}$ and coarser than $\mathcal{B}_{AP}$

# Partitioning splitter algorithms

*Kanellakis-Smolka algorithm:*

initially: $\mathcal{B}_{\text{old}} = \mathcal{B} = \mathcal{B}_{AP}$

iteration: stabilization for each block in $\mathcal{B}_{\text{old}}$

loop invariant: $\mathcal{B}$ finer than $\mathcal{B}_{\text{old}}$ and coarser than $\mathcal{B}_{AP}$

*Paige-Tarjan algorithm:*

*Kanellakis-Smolka algorithm:*

initially: $\mathcal{B}_{\text{old}} = \mathcal{B} = \mathcal{B}_{AP}$

iteration: stabilization for each block in $\mathcal{B}_{\text{old}}$

loop invariant: $\mathcal{B}$ finer than $\mathcal{B}_{\text{old}}$ and coarser than $\mathcal{B}_{AP}$

*Paige-Tarjan algorithm:*

loop invariant:

(1) $\mathcal{B}$ finer than $\mathcal{B}_{\text{old}}$ and coarser than $\mathcal{B}_{AP}$

(2) $\mathcal{B}$ is stable for each block in $\mathcal{B}_{\text{old}}$

# Partitioning splitter algorithms



*Kanellakis-Smolka algorithm:*

initially: $\mathcal{B}_{\mathsf{old}} = \mathcal{B} = \mathcal{B}_{AP}$

iteration: stabilization for each block in $\mathcal{B}_{\mathsf{old}}$

loop invariant: $\mathcal{B}$ finer than $\mathcal{B}_{\mathsf{old}}$ and coarser than $\mathcal{B}_{AP}$

*Paige-Tarjan algorithm:*

loop invariant:

(1) $\mathcal{B}$ finer than $\mathcal{B}_{\mathsf{old}}$ and coarser than $\mathcal{B}_{AP}$

(2) $\mathcal{B}$ is stable for each block in $\mathcal{B}_{\mathsf{old}}$

iteration: ternary refinement operator

*Kanellakis-Smolka algorithm:*

initially: $\mathcal{B}_{\mathsf{old}} = \mathcal{B} = \mathcal{B}_{AP}$

iteration: stabilization for each block in $\mathcal{B}_{\mathsf{old}}$

loop invariant: $\mathcal{B}$ finer than $\mathcal{B}_{\mathsf{old}}$ and coarser than $\mathcal{B}_{AP}$

*Paige-Tarjan algorithm:*

loop invariant:

(1) $\mathcal{B}$ finer than $\mathcal{B}_{\mathsf{old}}$ and coarser than $\mathcal{B}_{AP}$

(2) $\mathcal{B}$ is stable for each block in $\mathcal{B}_{\mathsf{old}}$

iteration: ternary refinement operator

initially: $\mathcal{B}_{\mathsf{old}} = \{S\}$

# Partitioning splitter algorithms



*Kanellakis-Smolka algorithm:*

initially: $\mathcal{B}_{\mathsf{old}} = \mathcal{B} = \mathcal{B}_{AP}$

iteration: stabilization for each block in $\mathcal{B}_{\mathsf{old}}$

loop invariant: $\mathcal{B}$ finer than $\mathcal{B}_{\mathsf{old}}$ and coarser than $\mathcal{B}_{AP}$

*Paige-Tarjan algorithm:*

loop invariant:

(1) $\mathcal{B}$ finer than $\mathcal{B}_{\mathsf{old}}$ and coarser than $\mathcal{B}_{AP}$

(2) $\mathcal{B}$ is stable for each block in $\mathcal{B}_{\mathsf{old}}$

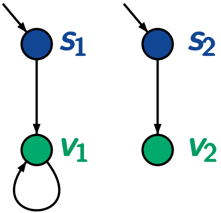
iteration: ternary refinement operator

initially: $\mathcal{B}_{\mathsf{old}} = \{S\}$, $\mathcal{B} = \mathit{Refine}(\mathcal{B}_{AP}, S)$

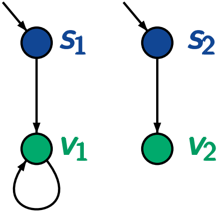
# $\mathcal{B}_{AP}$ is generally not stable w.r.t. $S$

# $\mathcal{B}_{AP}$ is generally not stable w.r.t. $S$





# $\mathcal{B}_{AP}$ is generally not stable w.r.t. $S$

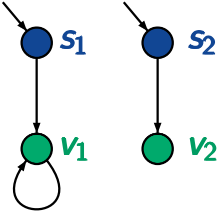



state space $S = \{s_1, s_2, v_1, v_2\}$

$$\mathcal{B}_{AP} = \Big\{ \{s_1, s_2\}, \{v_1, v_2\} \Big\}$$

# $\mathcal{B}_{AP}$ is generally not stable w.r.t. $S$



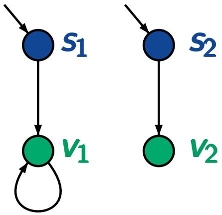



state space $S = \{s_1, s_2, v_1, v_2\}$

$$\mathcal{B}_{AP} = \Big\{ \{s_1, s_2\}, \{v_1, v_2\} \Big\}$$

$$\begin{aligned} Pre(S) &= \text{set of nonterminal states} \\ &= \{s_1, s_2, v_1\} \end{aligned}$$

# $\mathcal{B}_{AP}$ is generally not stable w.r.t. $S$




state space $S = \{s_1, s_2, v_1, v_2\}$

$\mathcal{B}_{AP} = \{\{s_1, s_2\}, \{v_1, v_2\}\}$

$Pre(S) =$ set of nonterminal states

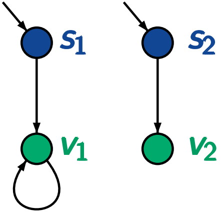
$= \{s_1, s_2, v_1\}$

$$\{v_1, v_2\} \cap Pre(S) = \{v_1\}$$

$$\{v_1, v_2\} \setminus Pre(S) = \{v_2\}$$

# $\mathcal{B}_{AP}$ is generally not stable w.r.t. $S$






state space $S = \{s_1, s_2, v_1, v_2\}$

$\mathcal{B}_{AP} = \big\{ \{s_1, s_2\}, \{v_1, v_2\} \big\}$

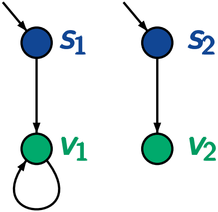
$Pre(S) =$ set of nonterminal states $= \{s_1, s_2, v_1\}$

$$\{v_1, v_2\} \cap Pre(S) = \{v_1\}$$
$$\{v_1, v_2\} \setminus Pre(S) = \{v_2\}$$

initial partition of Paige/Tarjan algorithm:

$$Refine(\mathcal{B}_{AP}, S)$$

# $\mathcal{B}_{AP}$ is generally not stable w.r.t. $S$




state space $S = \{s_1, s_2, v_1, v_2\}$

$$\mathcal{B}_{AP} = \Big\{ \{s_1, s_2\}, \{v_1, v_2\} \Big\}$$

$Pre(S)$ = set of nonterminal states $= \{s_1, s_2, v_1\}$

$$\{v_1, v_2\} \cap Pre(S) = \{v_1\}$$
$$\{v_1, v_2\} \setminus Pre(S) = \{v_2\}$$

initial partition of Paige/Tarjan algorithm:

$$Refine(\mathcal{B}_{AP}, S) = \Big\{ \{s_1, s_2\}, \{v_1\}, \{v_2\} \Big\}$$

# Paige-Tarjan algorithm

$\mathcal{B}_{\mathsf{old}} := \{S\}$; $\mathcal{B} := \mathit{Refine}(\mathcal{B}_{AP}, S)$;

`WHILE` $\mathcal{B} \neq \mathcal{B}_{\mathsf{old}}$ `DO`

`OD`

$\mathcal{B}_{\mathsf{old}} := \{S\}$; $\mathcal{B} := \mathit{Refine}(\mathcal{B}_{AP}, S)$;

```
WHILE B ≠ B_old DO
    select a block C' ∈ B_old \ B;
OD
```

$\mathcal{B}_{\mathsf{old}} := \{S\}$; $\mathcal{B} := \mathit{Refine}(\mathcal{B}_{AP}, S)$;

WHILE $\mathcal{B} \neq \mathcal{B}_{\mathsf{old}}$ DO

select a block $C' \in \mathcal{B}_{\mathsf{old}} \setminus \mathcal{B}$;

select a block $C \in \mathcal{B}$ with $C \subseteq C'$

OD

$\mathcal{B}_{\text{old}} := \{S\}$; $\mathcal{B} := \mathit{Refine}(\mathcal{B}_{AP}, S)$;

WHILE $\mathcal{B} \neq \mathcal{B}_{\text{old}}$ DO

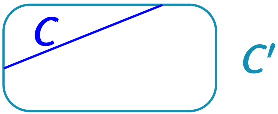
select a block $C' \in \mathcal{B}_{\text{old}} \setminus \mathcal{B}$;

select a block $C \in \mathcal{B}$ with $C \subseteq C'$ and $|C| \leq |C'|/2$;

OD

# Paige-Tarjan algorithm



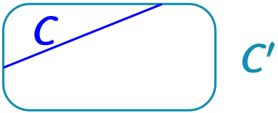



$\mathcal{B}_{\text{old}} := \{S\}$; $\mathcal{B} := \mathit{Refine}(\mathcal{B}_{AP}, S)$;

```
WHILE B ≠ B_old DO
   select a block C' ∈ B_old \ B;
   select a block C ∈ B with C ⊆ C' and |C| ≤ |C'|/2;






OD
```

# Paige-Tarjan algorithm






$\mathcal{B}_{\mathsf{old}} := \{S\}$; $\mathcal{B} := \mathit{Refine}(\mathcal{B}_{AP}, S)$;

WHILE $\mathcal{B} \neq \mathcal{B}_{\mathsf{old}}$ DO

  select a block $C' \in \mathcal{B}_{\mathsf{old}} \setminus \mathcal{B}$;

  select a block $C \in \mathcal{B}$ with $C \subseteq C'$ and $|C| \le |C'|/2$;

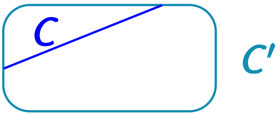
  refine $\mathcal{B}$
  w.r.t. $C$ and $C' \setminus C$

OD

# Paige-Tarjan algorithm



$\mathcal{B}_{\mathsf{old}} := \{S\}$; $\mathcal{B} := \mathit{Refine}(\mathcal{B}_{AP}, S)$;




WHILE $\mathcal{B} \neq \mathcal{B}_{\mathsf{old}}$ DO

select a block $C' \in \mathcal{B}_{\mathsf{old}} \setminus \mathcal{B}$;

select a block $C \in \mathcal{B}$ with $C \subseteq C'$ and $|C| \leq |C'|/2$;

$\mathcal{B} := \mathit{Refine}(\mathcal{B}, C)$

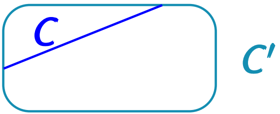
$\mathcal{B} := \mathit{Refine}(\mathcal{B}, C')$

refine $\mathcal{B}$
w.r.t. $C$ and $C' \setminus C$

OD

# Paige-Tarjan algorithm






$\mathcal{B}_{\text{old}} := \{S\}$; $\mathcal{B} := \mathit{Refine}(\mathcal{B}_{AP}, S)$;

```
WHILE 𝓑 ≠ 𝓑_old DO
```

select a block $C' \in \mathcal{B}_{\text{old}} \setminus \mathcal{B}$;

select a block $C \in \mathcal{B}$ with $C \subseteq C'$ and $|C| \leq |C'|/2$;

$\mathcal{B} := \mathit{Refine}(\mathcal{B}, C)$

$\mathcal{B} := \mathit{Refine}(\mathcal{B}, C')$

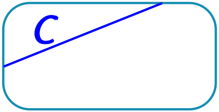
refine $\mathcal{B}$ simultaneously
w.r.t. $C$ and $C' \setminus C$

```
OD
```

# Paige-Tarjan algorithm



$\mathcal{B}_{\text{old}} := \{S\}$; $\mathcal{B} := \mathit{Refine}(\mathcal{B}_{AP}, S)$;




$C'$

```
WHILE
```
$\mathcal{B} \neq \mathcal{B}_{\text{old}}$ DO

select a block $C' \in \mathcal{B}_{\text{old}} \setminus \mathcal{B}$;

select a block $C \in \mathcal{B}$ with $C \subseteq C'$ and $|C| \leq |C'|/2$;

$\mathcal{B} := \mathit{Refine}(\mathcal{B}, C, C' \setminus C)$

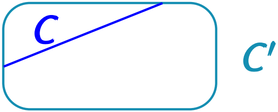
refine $\mathcal{B}$ simultaneously w.r.t. $C$ and $C' \setminus C$

OD

# Paige-Tarjan algorithm



$\mathcal{B}_{\text{old}} := \{S\}$; $\mathcal{B} := \mathit{Refine}(\mathcal{B}_{AP}, S)$;




```
WHILE  𝓑 ≠ 𝓑old  DO
```

select a block $C' \in \mathcal{B}_{\text{old}} \setminus \mathcal{B}$;

select a block $C \in \mathcal{B}$ with $C \subseteq C'$ and $|C| \leq |C'|/2$;

$\mathcal{B} := \mathit{Refine}(\mathcal{B}, C, C' \setminus C)$

refine $\mathcal{B}$ simultaneously w.r.t. $C$ and $C' \setminus C$

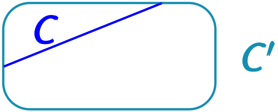
add $C$ and $C' \setminus C$ to $\mathcal{B}_{\text{old}}$

```
OD
```

# Paige-Tarjan algorithm



$\mathcal{B}_{\text{old}} := \{S\}$; $\mathcal{B} := \textit{Refine}(\mathcal{B}_{AP}, S)$;




WHILE $\mathcal{B} \neq \mathcal{B}_{\text{old}}$ DO

select a block $C' \in \mathcal{B}_{\text{old}} \setminus \mathcal{B}$;

select a block $C \in \mathcal{B}$ with $C \subseteq C'$ and $|C| \leq |C'|/2$;

$\mathcal{B} := \textit{Refine}(\mathcal{B}, C, C' \setminus C)$

refine $\mathcal{B}$ simultaneously w.r.t. $C$ and $C' \setminus C$

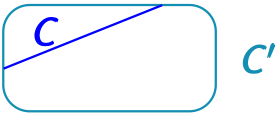
add $C$ and $C' \setminus C$ to $\mathcal{B}_{\text{old}}$ and remove $C'$ from $\mathcal{B}_{\text{old}}$

OD

# Paige-Tarjan algorithm



$\mathcal{B}_{\mathsf{old}} := \{S\}$; $\mathcal{B} := \mathit{Refine}(\mathcal{B}_{AP}, S)$;




WHILE $\mathcal{B} \neq \mathcal{B}_{\mathsf{old}}$ DO

select a block $C' \in \mathcal{B}_{\mathsf{old}} \setminus \mathcal{B}$;

select a block $C \in \mathcal{B}$ with $C \subseteq C'$ and $|C| \leq |C'|/2$;

$\mathcal{B} := \mathit{Refine}(\mathcal{B}, C, C' \setminus C)$

refine $\mathcal{B}$ simultaneously w.r.t. $C$ and $C' \setminus C$

add $C$ and $C' \setminus C$ to $\mathcal{B}_{\mathsf{old}}$ and remove $C'$ from $\mathcal{B}_{\mathsf{old}}$

OD

loop invariant: $\mathcal{B}$ is stable w.r.t. each block in $\mathcal{B}_{\mathsf{old}}$

## The ternary refinement operator

Let $\mathcal{B}$ be a partition and

- $C'$ a superblock of $\mathcal{B}$ s.t. $\mathcal{B}$ is stable w.r.t. $C'$

# The ternary refinement operator



Let $\mathcal{B}$ be a partition and

- $C'$ a superblock of $\mathcal{B}$ s.t. $\mathcal{B}$ is stable w.r.t. $C'$

Let $\mathcal{B}$ be a partition and

- $C'$ a superblock of $\mathcal{B}$ s.t. $\mathcal{B}$ is stable w.r.t. $C'$
- $C$ a block in $\mathcal{B}$ s.t. $C \subseteq C'$

# The ternary refinement operator

Let $\mathcal{B}$ be a partition and

- $C'$ a superblock of $\mathcal{B}$ s.t. $\mathcal{B}$ is stable w.r.t. $C'$
- $C$ a block in $\mathcal{B}$ s.t. $C \subseteq C'$

simultaneous refinement of $\mathcal{B}$ w.r.t. $C$ and $C' \setminus C$:

# The ternary refinement operator

Let $\mathcal{B}$ be a partition and

- $C'$ a superblock of $\mathcal{B}$ s.t. $\mathcal{B}$ is stable w.r.t. $C'$
- $C$ a block in $\mathcal{B}$ s.t. $C \subseteq C'$

simultaneous refinement of $\mathcal{B}$ w.r.t. $C$ and $C' \setminus C$:

$$\mathit{Refine}(\mathcal{B}, C, C' \setminus C) = \bigcup_{B \in \mathcal{B}} \mathit{Refine}(B, C, C' \setminus C)$$

Let $\mathcal{B}$ be a partition and

- $C'$ a superblock of $\mathcal{B}$ s.t. $\mathcal{B}$ is stable w.r.t. $C'$
- $C$ a block in $\mathcal{B}$ s.t. $C \subseteq C'$

simultaneous refinement of $\mathcal{B}$ w.r.t. $C$ and $C' \setminus C$:

$$\mathit{Refine}(\mathcal{B}, C, C' \setminus C) = \bigcup_{B \in \mathcal{B}} \mathit{Refine}(B, C, C' \setminus C)$$

where for block $B \subseteq \mathit{Pre}(C')$:

$$\mathit{Refine}(B, C, C' \setminus C) = \{B_1, B_2, B_3\} \setminus \{\varnothing\}$$

# The ternary refinement operator

Let $\mathcal{B}$ be a partition and

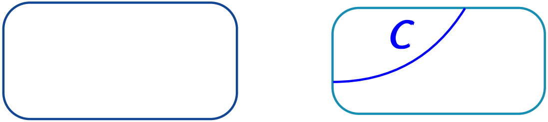
- $C'$ a superblock of $\mathcal{B}$ s.t. $\mathcal{B}$ is stable w.r.t. $C'$
- $C$ a block in $\mathcal{B}$ s.t. $C \subseteq C'$

simultaneous refinement of $\mathcal{B}$ w.r.t. $C$ and $C' \setminus C$:

$$Refine(\mathcal{B}, C, C' \setminus C) = \bigcup_{B \in \mathcal{B}} Refine(B, C, C' \setminus C)$$

where for block $B \subseteq Pre(C')$:

$$Refine(B, C, C' \setminus C) = \{B_1, B_2, B_3\} \setminus \{\varnothing\}$$




block $B$

superblock $C'$

# The ternary refinement operator

Let $\mathcal{B}$ be a partition and

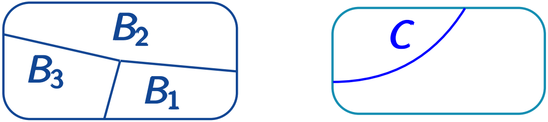
- $C'$ a superblock of $\mathcal{B}$ s.t. $\mathcal{B}$ is stable w.r.t. $C'$
- $C$ a block in $\mathcal{B}$ s.t. $C \subseteq C'$

simultaneous refinement of $\mathcal{B}$ w.r.t. $C$ and $C' \setminus C$:

$$Refine(\mathcal{B}, C, C' \setminus C) = \bigcup_{B \in \mathcal{B}} Refine(B, C, C' \setminus C)$$

where for block $B \subseteq Pre(C')$:

$$Refine(B, C, C' \setminus C) = \{B_1, B_2, B_3\} \setminus \{\varnothing\}$$



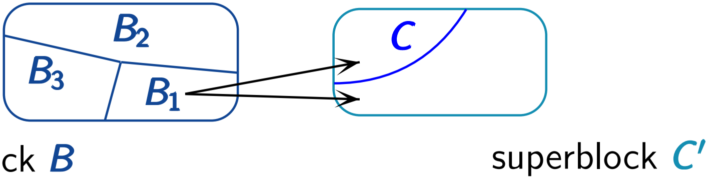

block $B$ superblock $C'$

simultaneous refinement of $\mathcal{B}$ w.r.t. $C$ and $C' \setminus C$:

$$Refine(\mathcal{B}, C, C' \setminus C) = \bigcup_{B \in \mathcal{B}} Refine(B, C, C' \setminus C)$$

where for block $B \subseteq Pre(C')$:

$$Refine(B, C, C' \setminus C) = \{B_1, B_2, B_3\} \setminus \{\varnothing\}$$



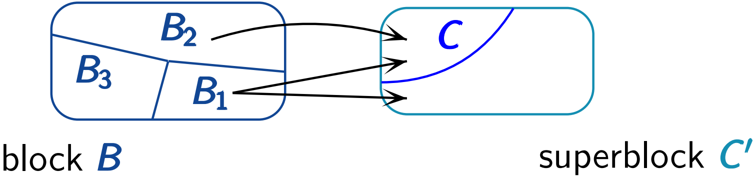

$$B_1 = B \cap Pre(C) \cap Pre(C' \setminus C)$$

simultaneous refinement of $\mathcal{B}$ w.r.t. $C$ and $C' \setminus C$:

$$Refine(\mathcal{B}, C, C' \setminus C) = \bigcup_{B \in \mathcal{B}} Refine(B, C, C' \setminus C)$$

where for block $B \subseteq Pre(C')$:

$$Refine(B, C, C' \setminus C) = \{B_1, B_2, B_3\} \setminus \{\varnothing\}$$




$$\begin{aligned} B_1 &= B \cap Pre(C) \cap Pre(C' \setminus C) \\ B_2 &= (B \cap Pre(C)) \setminus Pre(C' \setminus C) \end{aligned}$$

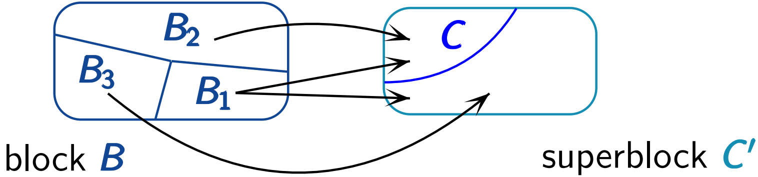
# The ternary refinement operator

simultaneous refinement of $\mathcal{B}$ w.r.t. $C$ and $C' \setminus C$:

$$Refine(\mathcal{B}, C, C' \setminus C) = \bigcup_{B \in \mathcal{B}} Refine(B, C, C' \setminus C)$$

where for block $B \subseteq Pre(C')$:

$$Refine(B, C, C' \setminus C) = \{B_1, B_2, B_3\} \setminus \{\varnothing\}$$




$$\boxed{\begin{aligned} B_1 &= B \cap Pre(C) \cap Pre(C' \setminus C) \\ B_2 &= (B \cap Pre(C)) \setminus Pre(C' \setminus C) \\ B_3 &= (B \cap Pre(C' \setminus C)) \setminus Pre(C) \end{aligned}}$$

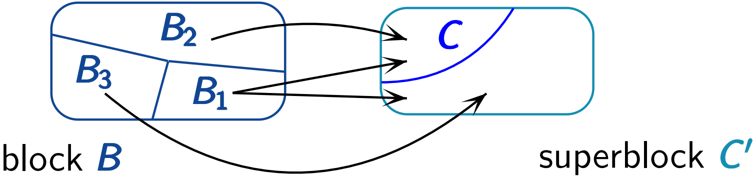
# The ternary refinement operator

simultaneous refinement of $\mathcal{B}$ w.r.t. $C$ and $C' \setminus C$:

$$Refine(\mathcal{B}, C, C' \setminus C) = \bigcup_{B \in \mathcal{B}} Refine(B, C, C' \setminus C)$$

where for block $B \subseteq Pre(C')$:

$$Refine(B, C, C' \setminus C) = \{B_1, B_2, B_3\} \setminus \{\varnothing\}$$




for block $B$ with $B \cap Pre(C') = \varnothing$:

$$Refine(B, C, C' \setminus C) = \{B\}$$

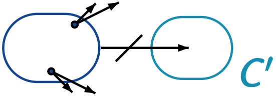
Suppose that for all blocks $B \in \mathcal{B}$:

$$B \subseteq Pre(C') \quad \text{or} \quad B \cap Pre(C') = \varnothing$$

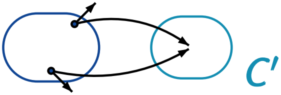
# Stability of $\mathit{Refine}(\mathcal{B}, C, C' \setminus C)$

Suppose that for all blocks $B \in \mathcal{B}$:

$$B \subseteq \mathit{Pre}(C') \quad \text{or} \quad B \cap \mathit{Pre}(C') = \varnothing$$

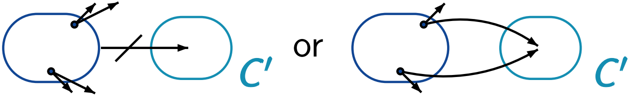



or

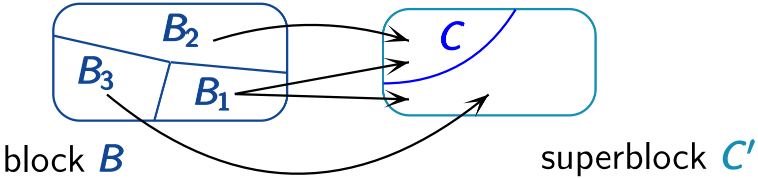


Suppose that for all blocks $B \in \mathcal{B}$:

$$B \subseteq \mathit{Pre}(C') \quad \text{or} \quad B \cap \mathit{Pre}(C') = \varnothing$$

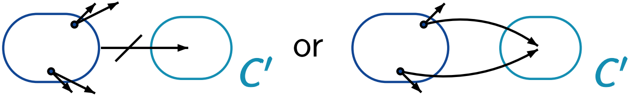



Then the new blocks $B_1, B_2, B_3$ in *Refine*$(B, C, C' \setminus C)$ are stable w.r.t. the superblocks $C$ and $C' \setminus C$.



Suppose that for all blocks $B \in \mathcal{B}$:

$$B \subseteq Pre(C') \quad \text{or} \quad B \cap Pre(C') = \varnothing$$



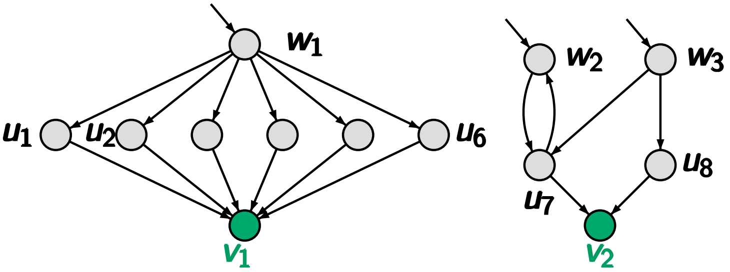

Then the new blocks $B_1, B_2, B_3$ in $Refine(B, C, C' \setminus C)$ are stable w.r.t. the superblocks $C$ and $C' \setminus C$.

If $\mathcal{B}$ is stable w.r.t. all blocks in $\mathcal{B}_{old}$ and $C' \in \mathcal{B}_{old}$, $C \in \mathcal{B}$ s.t. $C \subsetneq C'$ then $Refine(\mathcal{B}, C, C' \setminus C)$ is stable w.r.t. all blocks in the partition

$$(\mathcal{B}_{old} \setminus \{C'\}) \cup \{C, C' \setminus C\}$$

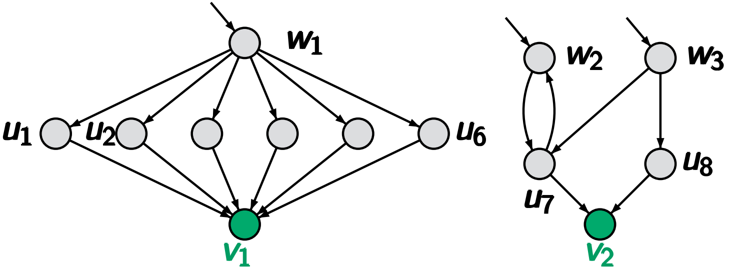
# Example: Paige-Tarjan algorithm






$AP = \{green, gray\}, \quad \mathcal{B}_{old} = \{S\}$

# Example: Paige-Tarjan algorithm

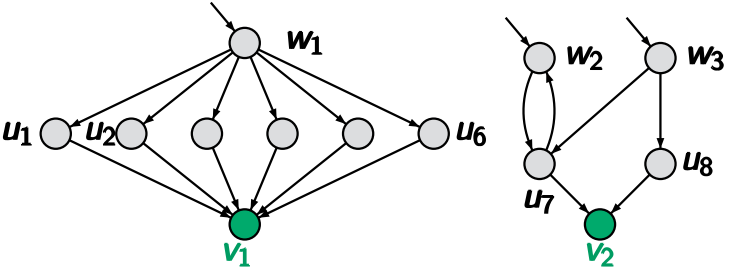



$AP = \{green, gray\}, \quad \mathcal{B}_{old} = \{S\}$

initial partition:

$$\begin{aligned}\mathcal{B}_0 &= Refine(\mathcal{B}_{AP}, S) = \mathcal{B}_{AP} \\ &= \big\{\{v_1, v_2\}, \{u_1, \ldots, u_8, w_1, w_2, w_3\}\big\}\end{aligned}$$

# Example: Paige-Tarjan algorithm




initially: $\mathcal{B}_{\text{old}} = \{S\}$

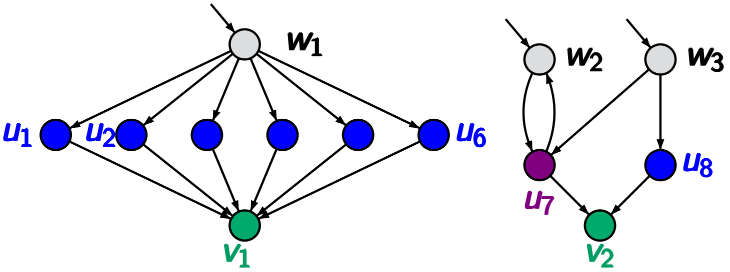
$\mathcal{B}_0 = \{\{v_1, v_2\}, \{u_1, \ldots, u_8, w_1, w_2, w_3\}\}$

first refinement step:

$Refine(\mathcal{B}_0, \{v_1, v_2\}, S \setminus \{v_1, v_2\})$

# Example: Paige-Tarjan algorithm






initially: $\mathcal{B}_{\text{old}} = \{S\}$

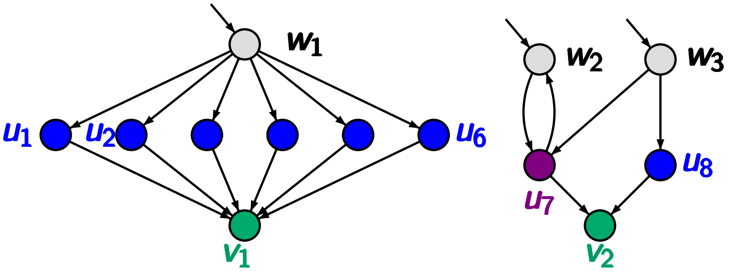
$\mathcal{B}_0 = \big\{\{v_1, v_2\}, \{u_1, \ldots, u_6, u_8, u_7, w_1, w_2, w_3\}\big\}$

first refinement step:

$$Refine(\mathcal{B}_0, \{v_1, v_2\}, S \setminus \{v_1, v_2\}) =$$

$$\mathcal{B}_1 = \big\{\{v_1, v_2\}, \{u_1, \ldots, u_6, u_8\}, \{u_7\}, \{w_1, w_2, w_3\}\big\}$$

# Example: Paige-Tarjan algorithm




initially: $\mathcal{B}_{old} = \{S\}$

$\mathcal{B}_0 = \{\{v_1, v_2\}, \{u_1, \ldots, u_6, u_8, u_7, w_1, w_2, w_3\}\}$

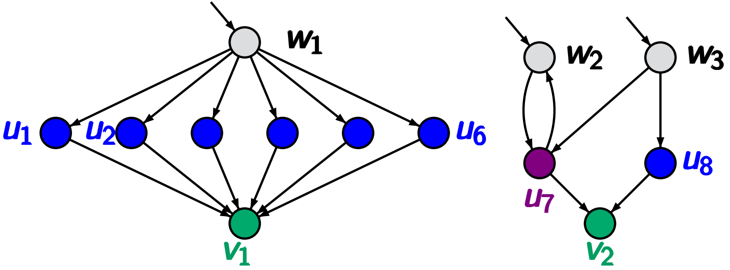
first refinement step:

$$Refine(\mathcal{B}_0, \{v_1, v_2\}, S \setminus \{v_1, v_2\}) =$$

$$\mathcal{B}_1 = \{\{v_1, v_2\}, \{u_1, \ldots, u_6, u_8\}, \{u_7\}, \{w_1, w_2, w_3\}\}$$

$$\mathcal{B}_{old} = \{\{v_1, v_2\}, \{u_1, \ldots, u_6, u_8, u_7, w_1, w_2, w_3\}\}$$

# Example: Paige-Tarjan algorithm



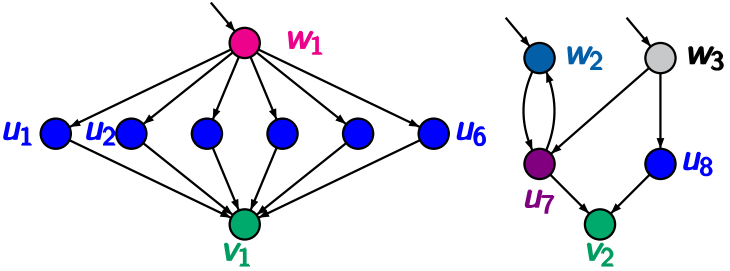

first refinement step:

$$\mathcal{B}_1 = \Big\{\{v_1, v_2\}, \{u_1, \ldots, u_6, u_8\}, \{u_7\}, \{w_1, w_2, w_3\}\Big\}$$
$$\mathcal{B}_{old} = \Big\{\{v_1, v_2\}, \{u_1, \ldots, u_6, u_8, u_7, w_1, w_2, w_3\}\Big\}$$

second refinement step:

$$\mathit{Refine}(\mathcal{B}_1, ?, ?)$$

# Example: Paige-Tarjan algorithm




first refinement step:

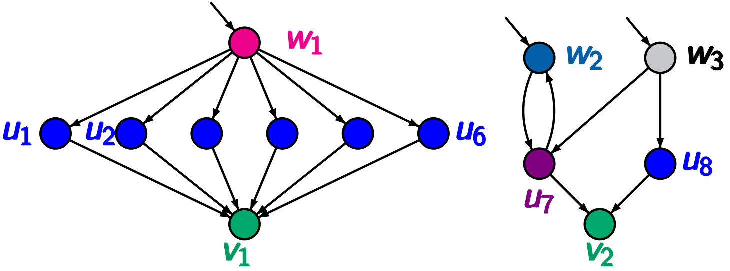
$$\mathcal{B}_1 = \bigl\{\{v_1, v_2\}, \{u_1, \ldots, u_6, u_8\}, \{u_7\}, \{w_1, w_2, w_3\}\bigr\}$$
$$\mathcal{B}_{old} = \bigl\{\{v_1, v_2\}, \{u_1, \ldots, u_6, u_8, u_7, w_1, w_2, w_3\}\bigr\}$$

second refinement step:

$$Refine(\mathcal{B}_1, \{u_7\}, \{u_1, \ldots, u_6, u_8, w_1, w_2, w_3\})$$

# Example: Paige-Tarjan algorithm






first refinement step:

$$\mathcal{B}_1 = \{\{v_1, v_2\}, \{u_1, \ldots, u_6, u_8\}, \{u_7\}, \{w_1, w_2, w_3\}\}$$
$$\mathcal{B}_{old} = \{\{v_1, v_2\}, \{u_1, \ldots, u_6, u_8, u_7, w_1, w_2, w_3\}\}$$

second refinement step:

$$Refine(\mathcal{B}_1, \{u_7\}, \{u_1, \ldots, u_6, u_8, w_1, w_2, w_3\})$$
$$= \{\{v_1, v_2\}, \{u_1, \ldots, u_6, u_8\}, \{u_7\}, \{w_1\}, \{w_2\}, \{w_3\}\}$$

# Paige-Tarjan algorithm



$\mathcal{B} := \mathit{Refine}(\mathcal{B}_{AP}, S)$; $\mathcal{B}_{\mathsf{old}} := \{S\}$;

`WHILE` $\mathcal{B} \neq \mathcal{B}_{\mathsf{old}}$ `DO`

  select $C' \in \mathcal{B}_{\mathsf{old}}$, $C \in \mathcal{B}$ s.t. $C \subseteq C'$, $|C| \leq |C'|/2$;

  add $C$ and $C' \setminus C$ to $\mathcal{B}_{\mathsf{old}}$ and remove $C'$ from $\mathcal{B}_{\mathsf{old}}$

  $\mathcal{B} := \mathit{Refine}(\mathcal{B}, C, C' \setminus C)$

`OD`

return $\mathcal{B}$

# Paige-Tarjan algorithm

```
B := Refine(B_AP, S);  B_old := {S};
WHILE B ≠ B_old DO
   select C' ∈ B_old, C ∈ B s.t. C ⊆ C', |C| ≤ |C'|/2;
   add C and C' \ C to B_old and remove C' from B_old
   B := Refine(B, C, C' \ C)
OD
return B
```

$\mathcal{B} := \mathit{Refine}(\mathcal{B}_{AP}, S)$; $\mathcal{B}_{\mathsf{old}} := \{S\}$;
WHILE $\mathcal{B} \neq \mathcal{B}_{\mathsf{old}}$ DO
  select $C' \in \mathcal{B}_{\mathsf{old}}$, $C \in \mathcal{B}$ s.t. $C \subseteq C'$, $|C| \leq |C'|/2$;
  add $C$ and $C' \setminus C$ to $\mathcal{B}_{\mathsf{old}}$ and remove $C'$ from $\mathcal{B}_{\mathsf{old}}$
  $\boxed{\mathcal{B} := \mathit{Refine}(\mathcal{B}, C, C' \setminus C)}$
OD
return $\mathcal{B}$

efficient implementation of $\mathit{Refine}(\mathcal{B}, C, \ldots)$ with time complexity $\mathcal{O}\big(|C| + |\mathit{Pre}(C)|\big)$

$\mathcal{B} := \mathit{Refine}(\mathcal{B}_{AP}, S)$; $\mathcal{B}_{\mathsf{old}} := \{S\}$;
WHILE $\mathcal{B} \neq \mathcal{B}_{\mathsf{old}}$ DO
 select $C' \in \mathcal{B}_{\mathsf{old}}$, $C \in \mathcal{B}$ s.t. $C \subseteq C'$, $|C| \leq |C'|/2$;
 add $C$ and $C' \setminus C$ to $\mathcal{B}_{\mathsf{old}}$ and remove $C'$ from $\mathcal{B}_{\mathsf{old}}$
 $\boxed{\mathcal{B} := \mathit{Refine}(\mathcal{B}, C, C' \setminus C)}$
OD
return $\mathcal{B}$

efficient implementation of $\mathit{Refine}(\mathcal{B}, C, \ldots)$ with time complexity $\mathcal{O}\big(|C| + |\mathit{Pre}(C)|\big)$ uses counters

$$\delta(s, D) = |\mathit{Post}(s) \cap D| \text{ for } D \in \mathcal{B}_{\mathsf{old}}$$

implementation of

$$\mathit{Refine}(\mathcal{B}, C, C' \setminus C) = \bigcup_{B \in \mathcal{B}} \mathit{Refine}(B, C, C' \setminus C)$$

using counters $\delta(s, D) = |\mathit{Post}(s) \cap D|$

# Details of the Paige-Tarjan algorithm

implementation of

$$Refine(\mathcal{B}, C, C' \setminus C) = \bigcup_{B \in \mathcal{B}} Refine(B, C, C' \setminus C)$$

using counters $\delta(s, D) = |Post(s) \cap D|$

$D \in \mathcal{B}_{\mathsf{old}}$

# Details of the Paige-Tarjan algorithm

implementation of

$$Refine(\mathcal{B}, C, C' \setminus C) = \bigcup_{B \in \mathcal{B}} Refine(B, C, C' \setminus C)$$

using counters $\delta(s, D) = |Post(s) \cap D|$

$s \in Pre(D)$ $\quad$ $D \in \mathcal{B}_{\text{old}}$

implementation of

$$Refine(\mathcal{B}, C, C' \setminus C) = \bigcup_{B \in \mathcal{B}} Refine(B, C, C' \setminus C)$$

using counters $\delta(s, D) = |Post(s) \cap D|$

$s \in Pre(D)$ $\qquad$ $D \in \mathcal{B}_{\text{old}}$

*step 1:* compute $\delta(\ldots)$ for the new blocks $C$ and $C' \setminus C$ in $\mathcal{B}_{\text{old}}$

implementation of

$$Refine(\mathcal{B}, C, C' \setminus C) = \bigcup_{B \in \mathcal{B}} Refine(B, C, C' \setminus C)$$

using counters $\delta(s, D) = |Post(s) \cap D|$

$s \in Pre(D)$ $\quad$ $D \in \mathcal{B}_{\text{old}}$

*step 1:* compute $\delta(\ldots)$ for the new blocks $C$ and $C' \setminus C$ in $\mathcal{B}_{\text{old}}$

*step 2*: compute $Refine(B, C, C' \setminus C)$ for all $B \in \mathcal{B}$

# Implementation of $\mathit{Refine}(\mathcal{B}, C, C' \setminus C)$



*step 1:* compute $\delta(s, C)$, $\delta(s, C' \setminus C)$

*step 2*: compute $\mathit{Refine}(B, C, C' \setminus C)$ for all $B \in \mathcal{B}$

# Implementation of $Refine(\mathcal{B}, C, C' \setminus C)$ 

*step 1:* compute $\delta(s, C)$, $\delta(s, C' \setminus C) \longleftarrow \boxed{\text{for } s \in Pre(C')}$

*step 2*: compute $Refine(B, C, C' \setminus C)$ for all $B \in \mathcal{B}$

# Implementation of $Refine(\mathcal{B}, C, C' \setminus C)$



*step 1:* compute $\delta(s, C)$, $\delta(s, C' \setminus C)$ $\longleftarrow$ $\boxed{\text{for } s \in Pre(C')}$

$$\delta(s, C) \qquad = |Post(s) \cap C|$$

$$\delta(s, C' \setminus C) = |Post(s) \cap (C' \setminus C)|$$

*step 2*: compute $Refine(B, C, C' \setminus C)$ for all $B \in \mathcal{B}$

# Implementation of $\mathit{Refine}(\mathcal{B}, C, C' \setminus C)$



*step 1:* compute $\delta(s, C)$, $\delta(s, C' \setminus C) \longleftarrow \boxed{\text{for } s \in \mathit{Pre}(C')}$

$$\delta(s, C) = |\mathit{Post}(s) \cap C|$$

$$\delta(s, C' \setminus C) = |\mathit{Post}(s) \cap (C' \setminus C)|$$

*step 2:* compute $\mathit{Refine}(B, C, C' \setminus C)$ for all $B \in \mathcal{B}$

for $B \in \mathcal{B}$ with $B \cap \mathit{Pre}(C') = \varnothing$ we have:

$$\mathit{Refine}(B, C, C' \setminus C) = \{B\}$$

# Implementation of $Refine(\mathcal{B}, C, C' \setminus C)$



*step 1:* compute $\delta(s, C)$, $\delta(s, C' \setminus C)$ $\longleftarrow$ $\boxed{\text{for } s \in Pre(C')}$

$$\delta(s, C) \quad = |Post(s) \cap C|$$
$$\delta(s, C' \setminus C) = |Post(s) \cap (C' \setminus C)|$$

*step 2:* compute $Refine(B, C, C' \setminus C)$ for all $B \in \mathcal{B}$

for $B \in \mathcal{B}$ with $B \subseteq Pre(C')$:

# Implementation of $Refine(\mathcal{B}, C, C' \setminus C)$



*step 1:* compute $\delta(s, C)$, $\delta(s, C' \setminus C)$ $\longleftarrow$ $\boxed{\text{for } s \in Pre(C')}$

$$\delta(s, C) = |Post(s) \cap C|$$

$$\delta(s, C' \setminus C) = |Post(s) \cap (C' \setminus C)|$$

*step 2:* compute $Refine(B, C, C' \setminus C)$ for all $B \in \mathcal{B}$

for $B \in \mathcal{B}$ with $B \subseteq Pre(C')$:

$$Refine(B, C, C' \setminus C) = \{B_1, B_2, B_3\} \setminus \{\varnothing\}$$

# Implementation of $Refine(\mathcal{B}, C, C' \setminus C)$ 

*step 1:* compute $\delta(s, C)$, $\delta(s, C' \setminus C)$ $\longleftarrow$ $\boxed{\text{for } s \in Pre(C')}$

$$\delta(s, C) = |Post(s) \cap C|$$

$$\delta(s, C' \setminus C) = |Post(s) \cap (C' \setminus C)|$$

*step 2:* compute $Refine(B, C, C' \setminus C)$ for all $B \in \mathcal{B}$

for $B \in \mathcal{B}$ with $B \subseteq Pre(C')$:

$$Refine(B, C, C' \setminus C) = \{B_1, B_2, B_3\} \setminus \{\varnothing\}$$

$$B_1 = B \cap Pre(C) \cap Pre(C' \setminus C)$$

$$B_2 = (B \cap Pre(C)) \setminus Pre(C' \setminus C)$$

$$B_3 = (B \cap Pre(C' \setminus C)) \setminus Pre(C)$$

# Implementation of $Refine(\mathcal{B}, C, C' \setminus C)$



*step 1:* compute $\delta(s, C)$, $\delta(s, C' \setminus C)$ $\longleftarrow$ $\boxed{\text{for } s \in Pre(C')}$

$$\delta(s, C) = |Post(s) \cap C|$$

$$\delta(s, C' \setminus C) = |Post(s) \cap (C' \setminus C)|$$

*step 2:* compute $Refine(B, C, C' \setminus C)$ for all $B \in \mathcal{B}$

for $B \in \mathcal{B}$ with $B \subseteq Pre(C')$:

$$Refine(B, C, C' \setminus C) = \{B_1, B_2, B_3\} \setminus \{\varnothing\}$$

$$B_1 = \{s \in B : \delta(s, C) > 0, \delta(s, C' \setminus C) > 0\}$$

$$B_2 = (B \cap Pre(C)) \setminus Pre(C' \setminus C)$$

$$B_3 = (B \cap Pre(C' \setminus C)) \setminus Pre(C)$$

# Implementation of $Refine(\mathcal{B}, C, C' \setminus C)$



*step 1:* compute $\delta(s, C)$, $\delta(s, C' \setminus C)$ $\longleftarrow$ $\boxed{\text{for } s \in Pre(C')}$

$$\delta(s, C) \qquad = |Post(s) \cap C|$$

$$\delta(s, C' \setminus C) = |Post(s) \cap (C' \setminus C)|$$

*step 2:* compute $Refine(B, C, C' \setminus C)$ for all $B \in \mathcal{B}$

for $B \in \mathcal{B}$ with $B \subseteq Pre(C')$:

$$Refine(B, C, C' \setminus C) = \{B_1, B_2, B_3\} \setminus \{\varnothing\}$$

$$B_1 = \{s \in B : \delta(s, C) > 0, \delta(s, C' \setminus C) > 0\}$$

$$B_2 = \{s \in B : \delta(s, C) > 0, \delta(s, C' \setminus C) = 0\}$$

$$B_3 = (B \cap Pre(C' \setminus C)) \setminus Pre(C)$$

# Implementation of $Refine(\mathcal{B}, C, C' \setminus C)$



*step 1:* compute $\delta(s, C)$, $\delta(s, C' \setminus C)$ $\longleftarrow$ $\boxed{\text{for } s \in Pre(C')}$

$$\delta(s, C) = |Post(s) \cap C|$$

$$\delta(s, C' \setminus C) = |Post(s) \cap (C' \setminus C)|$$

*step 2:* compute $Refine(B, C, C' \setminus C)$ for all $B \in \mathcal{B}$

for $B \in \mathcal{B}$ with $B \subseteq Pre(C')$:

$$Refine(B, C, C' \setminus C) = \{B_1, B_2, B_3\} \setminus \{\varnothing\}$$

$$B_1 = \{s \in B : \delta(s, C) > 0, \delta(s, C' \setminus C) > 0\}$$

$$B_2 = \{s \in B : \delta(s, C) > 0, \delta(s, C' \setminus C) = 0\}$$

$$B_3 = \{s \in B : \delta(s, C) = 0, \delta(s, C' \setminus C) > 0\}$$

# Paige-Tarjan algorithm

$\mathcal{B} := \mathit{Refine}(\mathcal{B}_{\mathsf{AP}}, S)$; $\mathcal{B}_{\mathsf{old}} := \{S\}$;

```
WHILE  B ≠ B_old DO
  select C' ∈ B_old, C ∈ B s.t. C ⊆ C', |C| ≤ |C'|/2;
  add C and C'\C to B_old and remove C' from B_old

  B := Refine(B, C, C' \ C)
OD
```

$\texttt{WHILE}\ \ \mathcal{B} \neq \mathcal{B}_{\mathsf{old}}\ \texttt{DO}$

select $C' \in \mathcal{B}_{\mathsf{old}}$, $C \in \mathcal{B}$ s.t. $C \subseteq C'$, $|C| \leq |C'|/2$;

add $C$ and $C' \setminus C$ to $\mathcal{B}_{\mathsf{old}}$ and remove $C'$ from $\mathcal{B}_{\mathsf{old}}$

$\mathcal{B} := \mathit{Refine}(\mathcal{B}, C, C' \setminus C)$

$\texttt{OD}$

# Paige-Tarjan algorithm



$\mathcal{B} := \mathit{Refine}(\mathcal{B}_{\mathsf{AP}}, S)$; $\mathcal{B}_{\mathsf{old}} := \{S\}$;

`FOR ALL` $s \in S$ `DO` $\delta(s, S) := |\mathit{Post}(s)|$ `OD`

`WHILE` $\mathcal{B} \neq \mathcal{B}_{\mathsf{old}}$ `DO`

select $C' \in \mathcal{B}_{\mathsf{old}}$, $C \in \mathcal{B}$ s.t. $C \subseteq C'$, $|C| \leq |C'|/2$;

add $C$ and $C' \backslash C$ to $\mathcal{B}_{\mathsf{old}}$ and remove $C'$ from $\mathcal{B}_{\mathsf{old}}$

$\mathcal{B} := \mathit{Refine}(\mathcal{B}, C, C' \setminus C)$

`OD`

$\mathcal{B} := \mathit{Refine}(\mathcal{B}_{\mathrm{AP}}, S)$; $\mathcal{B}_{\mathrm{old}} := \{S\}$;

FOR ALL $s \in S$ DO $\delta(s, S) := |\mathit{Post}(s)|$ OD

WHILE $\mathcal{B} \neq \mathcal{B}_{\mathrm{old}}$ DO

select $C' \in \mathcal{B}_{\mathrm{old}}$, $C \in \mathcal{B}$ s.t. $C \subseteq C'$, $|C| \leq |C'|/2$;

add $C$ and $C' \backslash C$ to $\mathcal{B}_{\mathrm{old}}$ and remove $C'$ from $\mathcal{B}_{\mathrm{old}}$

FOR ALL $s \in \mathit{Pre}(C)$ DO $\delta(s, C) := 0$ OD

FOR ALL $s' \in C$ DO

FOR ALL $s \in \mathit{Pre}(s')$ DO $\delta(s, C) := \delta(s, C) + 1$ OD

OD

$\mathcal{B} := \mathit{Refine}(\mathcal{B}, C, C' \setminus C)$

OD

$\mathcal{B} := Refine(\mathcal{B}_{AP}, S)$; $\mathcal{B}_{old} := \{S\}$;

FOR ALL $s \in S$ DO $\delta(s, S) := |Post(s)|$ OD

WHILE $\mathcal{B} \neq \mathcal{B}_{old}$ DO

  select $C' \in \mathcal{B}_{old}$, $C \in \mathcal{B}$ s.t. $C \subseteq C'$, $|C| \leq |C'|/2$;

  add $C$ and $C' \backslash C$ to $\mathcal{B}_{old}$ and remove $C'$ from $\mathcal{B}_{old}$

    FOR ALL $s \in Pre(C)$ DO $\delta(s, C) := \mathbf{0}$ OD

    FOR ALL $s' \in C$ DO

      FOR ALL $s \in Pre(s')$ DO $\delta(s, C) := \delta(s, C) + \mathbf{1}$ OD

    OD

    FOR ALL $s \in Pre(C)$ DO

      $\delta(s, C' \backslash C) := \delta(s, C') - \delta(s, C)$ OD

  $\mathcal{B} := Refine(\mathcal{B}, C, C' \setminus C)$

OD

# Complexity of the Paige-Tarjan algorithm PARTSPLITALG5.3-26A

# Complexity of the Paige-Tarjan algorithm 

let $\mathcal{T} = (S, Act, \to, S_0, AP, L)$ be a finite TS

$$n = \#\ \text{states} \qquad = |S|$$

$$m = \#\ \text{transitions}$$

let $\mathcal{T} = (S, Act, \rightarrow, S_0, AP, L)$ be a finite TS

$$n = \#\ \text{states} = |S|$$

$$m = \#\ \text{transitions} = \sum_{s \in S} |Pre(s)|$$

# Complexity of the Paige-Tarjan algorithm 

let $\mathcal{T} = (S, Act, \to, S_0, AP, L)$ be a finite TS

$$n = \#\text{ states} = |S|$$
$$m = \#\text{ transitions} = \sum_{s \in S} |Pre(s)|$$

in what follows, we suppose $m \geq n$

# Complexity of the Paige-Tarjan algorithm

$\mathcal{B} := \mathit{Refine}(\mathcal{B}_{\mathsf{AP}}, S)$;

$\mathcal{B}_{\mathsf{old}} := \{S\}$;

`WHILE` $\mathcal{B} \neq \mathcal{B}_{\mathsf{old}}$ `DO`

select $C' \in \mathcal{B}_{\mathsf{old}}$, $C \in \mathcal{B}$ s.t.
$C \subseteq C'$ and $|C| \leq |C'|/2$;

add $C$ and $C' \setminus C$ to $\mathcal{B}_{\mathsf{old}}$ and
remove $C'$ from $\mathcal{B}_{\mathsf{old}}$

$\mathcal{B} := \mathit{Refine}(\mathcal{B}, C, C' \setminus C)$

`OD`

$\mathcal{B} := \mathit{Refine}(\mathcal{B}_{AP}, S)$; $\longleftarrow$ complexity: $\mathcal{O}(n \cdot |AP|)$

$\mathcal{B}_{old} := \{S\}$;

```
WHILE
```
$\mathcal{B} \neq \mathcal{B}_{old}$ DO

select $C' \in \mathcal{B}_{old}$, $C \in \mathcal{B}$ s.t.
$C \subseteq C'$ and $|C| \leq |C'|/2$;

add $C$ and $C' \setminus C$ to $\mathcal{B}_{old}$ and
remove $C'$ from $\mathcal{B}_{old}$

$\mathcal{B} := \mathit{Refine}(\mathcal{B}, C, C' \setminus C)$

OD

$\mathcal{B} := \mathit{Refine}(\mathcal{B}_{AP}, S)$; $\longleftarrow$ $\boxed{\text{complexity: } \mathcal{O}\big(n \cdot |AP|\big)}$

$\mathcal{B}_{old} := \{S\}$;

WHILE $\mathcal{B} \neq \mathcal{B}_{old}$ DO

select $C' \in \mathcal{B}_{old}$, $C \in \mathcal{B}$ s.t.
$C \subseteq C'$ and $|C| \leq |C'|/2$;

add $C$ and $C' \backslash C$ to $\mathcal{B}_{old}$ and
remove $C'$ from $\mathcal{B}_{old}$

$$\boxed{\mathcal{B} := \mathit{Refine}(\mathcal{B}, C, C' \setminus C)}$$

OD

$\mathcal{B} := \mathit{Refine}(\mathcal{B}_{AP}, S)$; $\longleftarrow$ complexity: $\mathcal{O}(n \cdot |AP|)$

$\mathcal{B}_{old} := \{S\}$;

WHILE $\mathcal{B} \neq \mathcal{B}_{old}$ DO

select $C' \in \mathcal{B}_{old}$, $C \in \mathcal{B}$ s.t.
$C \subseteq C'$ and $|C| \leq |C'|/2$;

add $C$ and $C' \setminus C$ to $\mathcal{B}_{old}$ and
remove $C'$ from $\mathcal{B}_{old}$

$\boxed{\mathcal{B} := \mathit{Refine}(\mathcal{B}, C, C' \setminus C)}$ $\longleftarrow$ time complexity: $\sum_{s' \in C} |\mathit{Pre}(s')| + 1$

OD

$\mathcal{B} := \mathit{Refine}(\mathcal{B}_{\mathsf{AP}}, S)$; $\longleftarrow$ complexity: $\mathcal{O}(n \cdot |AP|)$

$\mathcal{B}_{\mathsf{old}} := \{S\}$;

WHILE $\mathcal{B} \neq \mathcal{B}_{\mathsf{old}}$ DO

select $C' \in \mathcal{B}_{\mathsf{old}}$, $C \in \mathcal{B}$ s.t. $C \subseteq C'$ and $|C| \leq |C'|/2$;

add $C$ and $C' \setminus C$ to $\mathcal{B}_{\mathsf{old}}$ and remove $C'$ from $\mathcal{B}_{\mathsf{old}}$

$\boxed{\mathcal{B} := \mathit{Refine}(\mathcal{B}, C, C' \setminus C)}$ $\longleftarrow$ time complexity: $\mathcal{O}(|C| + |\mathit{Pre}(C)|)$

OD

$\mathcal{B} := Refine(\mathcal{B}_{AP}, S)$; $\longleftarrow$ complexity: $\mathcal{O}(n \cdot |AP|)$

$\mathcal{B}_{old} := \{S\}$;

WHILE $\mathcal{B} \neq \mathcal{B}_{old}$ DO

select $C' \in \mathcal{B}_{old}$, $C \in \mathcal{B}$ s.t. $C \subseteq C'$ and $|C| \leq |C'|/2$;

add $C$ and $C' \setminus C$ to $\mathcal{B}_{old}$ and remove $C'$ from $\mathcal{B}_{old}$

$\boxed{\mathcal{B} := Refine(\mathcal{B}, C, C' \setminus C)}$ $\longleftarrow$ time complexity: $\mathcal{O}(|C| + |Pre(C)|)$

OD

total cost for all refinement operations: $\mathcal{O}(m \cdot \log n)$