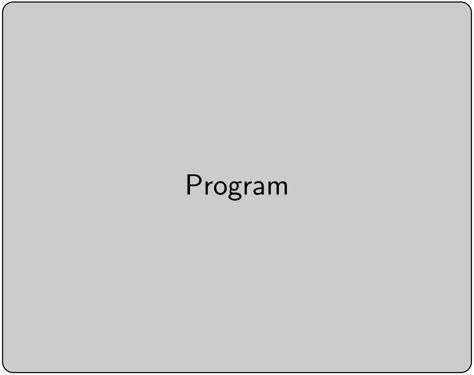
Control Flow Structures of Concurrent Programs are Higher Dimensional Mathematical Objects

Emmanuel Haucourt

Wednesday 13^{th} April 2016

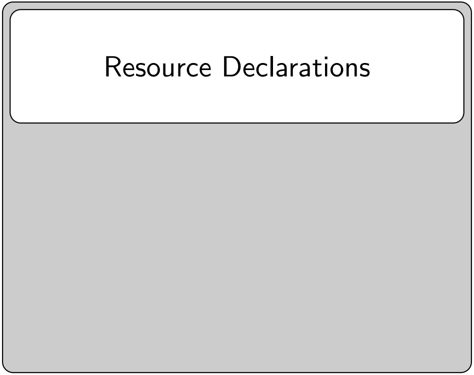
A Toy Language

from Dijkstra's "Cooperating Sequential Processes" paper



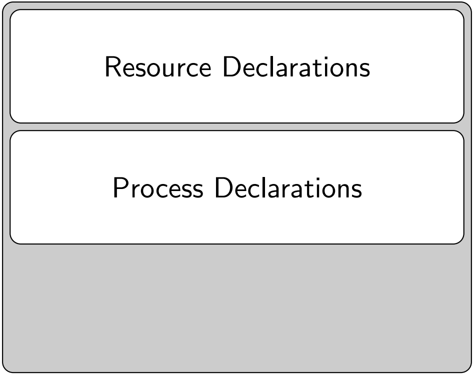
A Toy Language

from Dijkstra's "Cooperating Sequential Processes" paper



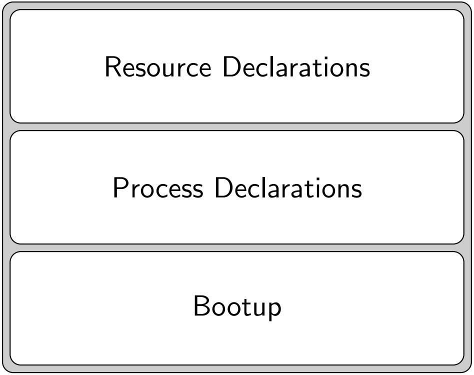
A Toy Language

from Dijkstra's "Cooperating Sequential Processes" paper



A Toy Language

from Dijkstra's "Cooperating Sequential Processes" paper



A Toy Language

Resource Declaration

A Toy Language

Resource Declaration

- sem: *<int>* *<set of identifiers>*

A Toy Language

Resource Declaration

- sem: *<int>* *<set of identifiers>*
- sync: *<int>* *<set of identifiers>*

A Toy Language

Resource Declaration

- sem: *<int>* *<set of identifiers>*
- sync: *<int>* *<set of identifiers>*
- var: *<identifier>* = *<constant>*

```
var:  x = 7
```

```
proc:
  p = ()+[x=1]+C(q)

proc:
  q = (x:=x/2  ; C(p))+[x % 2 = 0]+
      (x:=3*x+1; C(p))
```

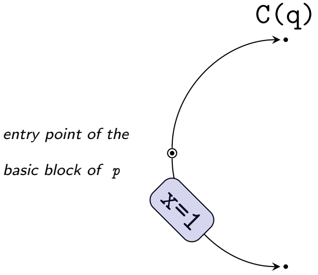
```
init:  p
```

Building the Control Flow Graph

of the Hasse-Syracuse algorithm

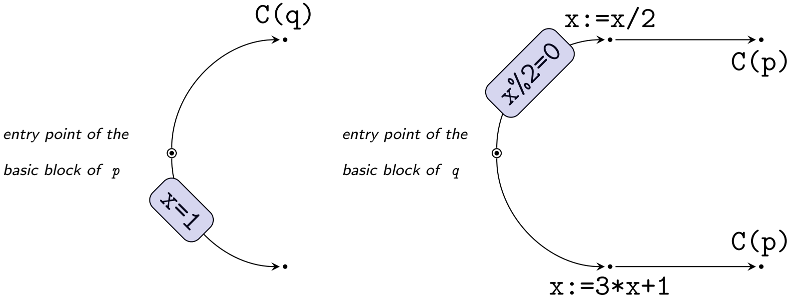
Building the Control Flow Graph

of the Hasse-Syracuse algorithm



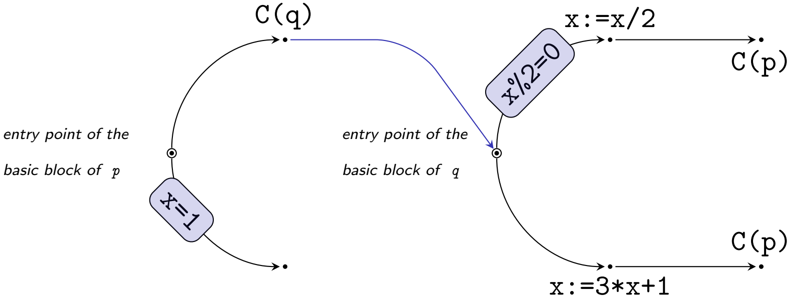
Building the Control Flow Graph

of the Hasse-Syracuse algorithm



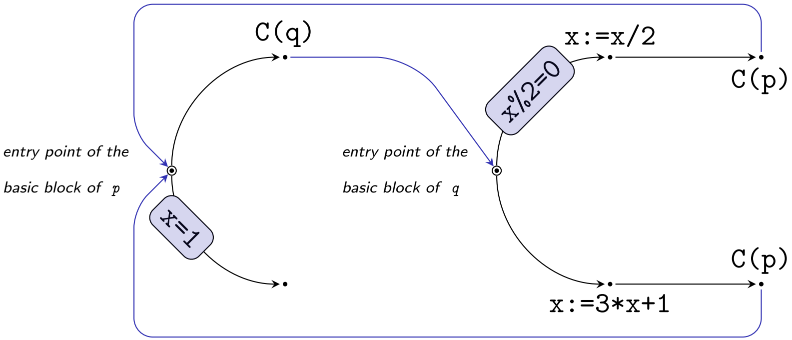
Building the Control Flow Graph

of the Hasse-Syracuse algorithm



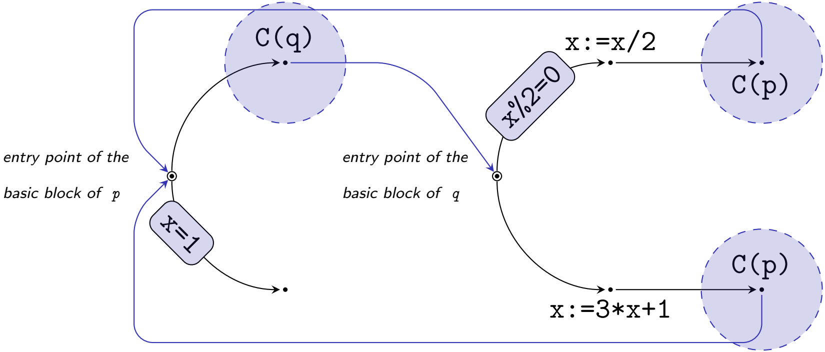
Building the Control Flow Graph

of the Hasse-Syracuse algorithm



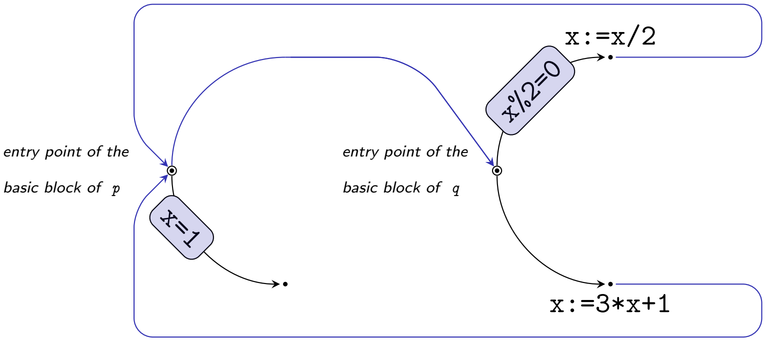
Building the Control Flow Graph

of the Hasse-Syracuse algorithm



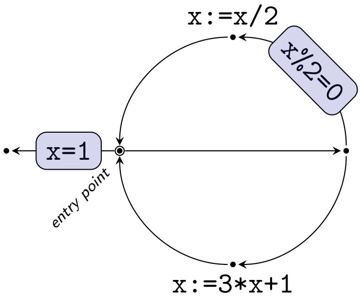
Building the Control Flow Graph

of the Hasse-Syracuse algorithm



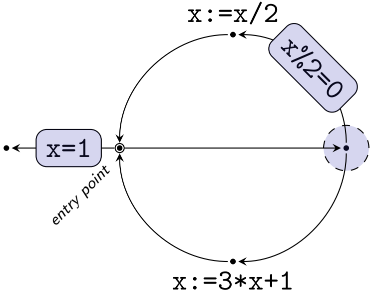
Reducing the Control Flow Graph

of the Hasse-Syracuse algorithm



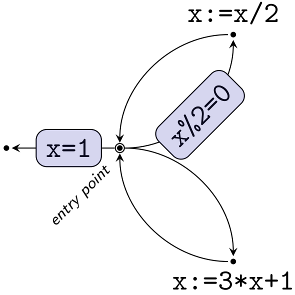
Reducing the Control Flow Graph

of the Hasse-Syracuse algorithm



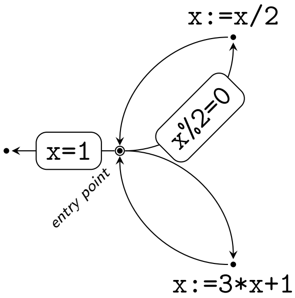
Reducing the Control Flow Graph

of the Hasse-Syracuse algorithm



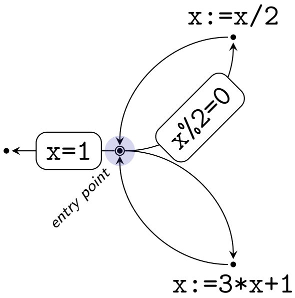
An Execution Trace

on a control flow graph



An Execution Trace

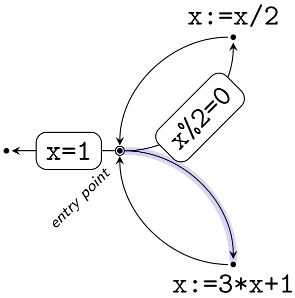
on a control flow graph



the current value of x is 7

An Execution Trace

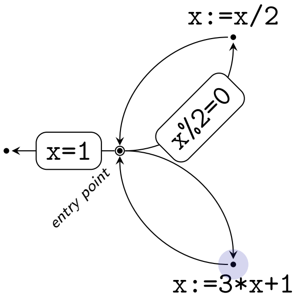
on a control flow graph



the current value of x is 7

An Execution Trace

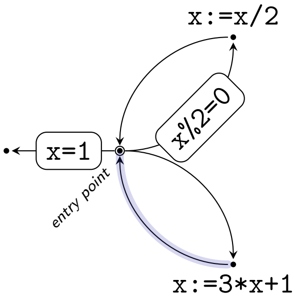
on a control flow graph



the current value of x is 22

An Execution Trace

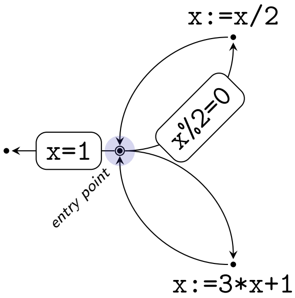
on a control flow graph



the current value of x is 22

An Execution Trace

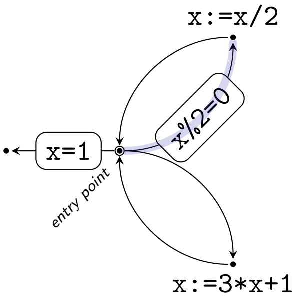
on a control flow graph



the current value of x is 22

An Execution Trace

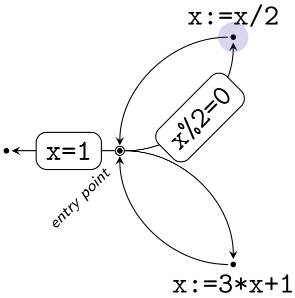
on a control flow graph



the current value of x is 22

An Execution Trace

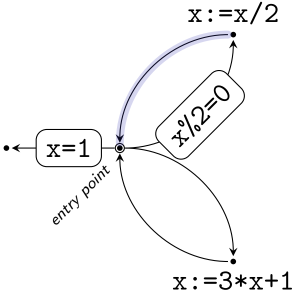
on a control flow graph



the current value of x is 11

An Execution Trace

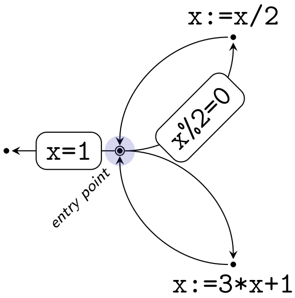
on a control flow graph



the current value of x is 11

An Execution Trace

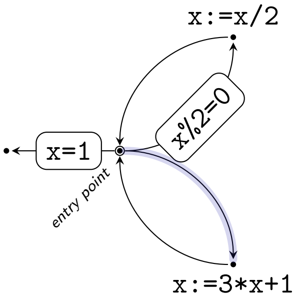
on a control flow graph



the current value of x is 11

An Execution Trace

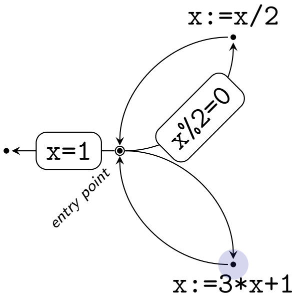
on a control flow graph



the current value of `x` is 11

An Execution Trace

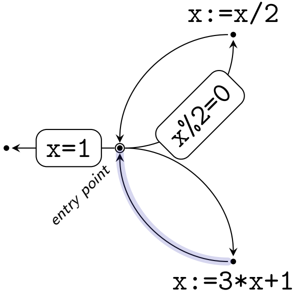
on a control flow graph



the current value of x is 34

An Execution Trace

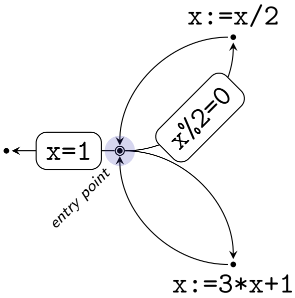
on a control flow graph



the current value of `x` is 34

An Execution Trace

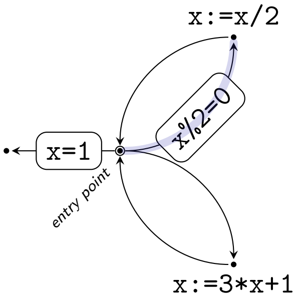
on a control flow graph



the current value of x is 34

An Execution Trace

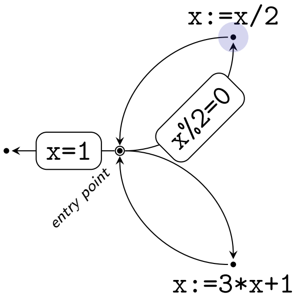
on a control flow graph



the current value of x is 34

An Execution Trace

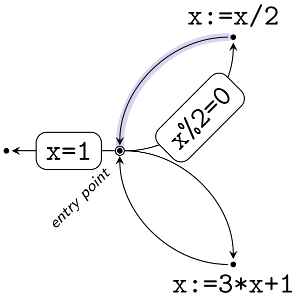
on a control flow graph



the current value of x is 17

An Execution Trace

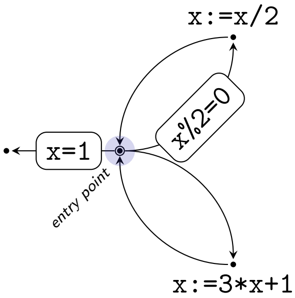
on a control flow graph



the current value of x is 17

An Execution Trace

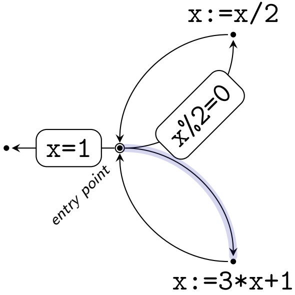
on a control flow graph



the current value of x is 17

An Execution Trace

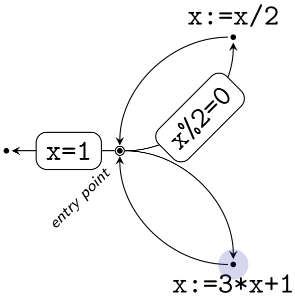
on a control flow graph



the current value of x is 17

An Execution Trace

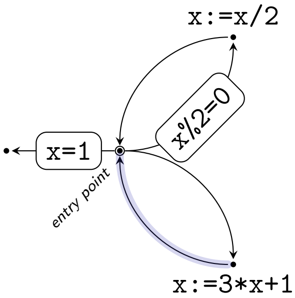
on a control flow graph



the current value of x is 52

An Execution Trace

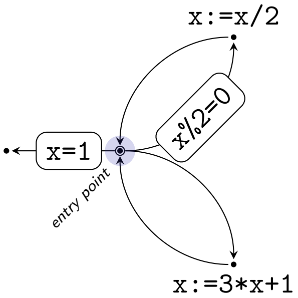
on a control flow graph



the current value of x is 52

An Execution Trace

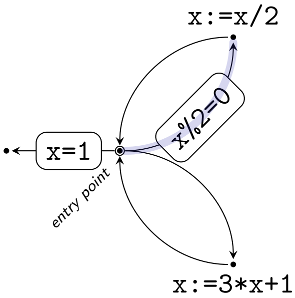
on a control flow graph



the current value of x is 52

An Execution Trace

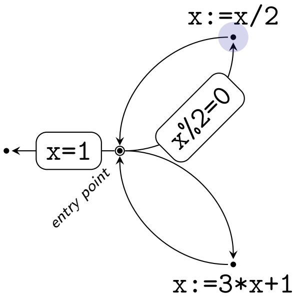
on a control flow graph



the current value of x is 52

An Execution Trace

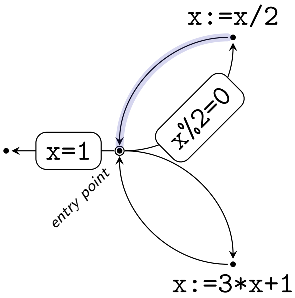
on a control flow graph



the current value of x is 26

An Execution Trace

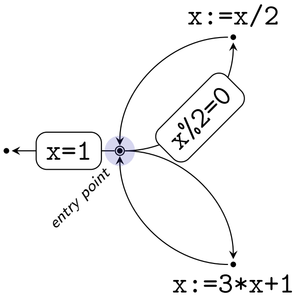
on a control flow graph



the current value of $\mathtt{x}$ is 26

An Execution Trace

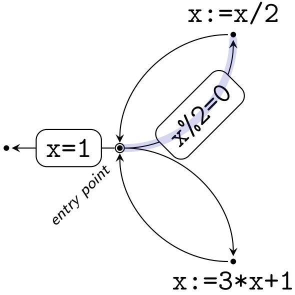
on a control flow graph



the current value of x is 26

An Execution Trace

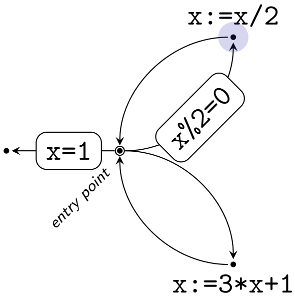
on a control flow graph



the current value of x is 26

An Execution Trace

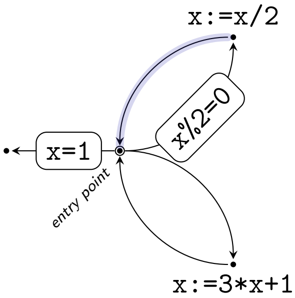
on a control flow graph



the current value of $\texttt{x}$ is 13

An Execution Trace

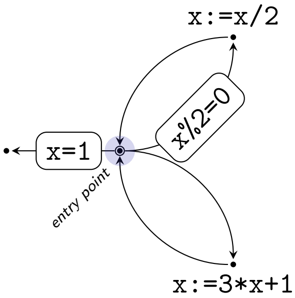
on a control flow graph



the current value of x is 13

An Execution Trace

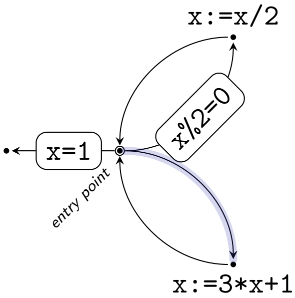
on a control flow graph



the current value of x is 13

An Execution Trace

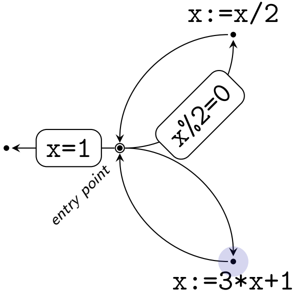
on a control flow graph



the current value of x is 13

An Execution Trace

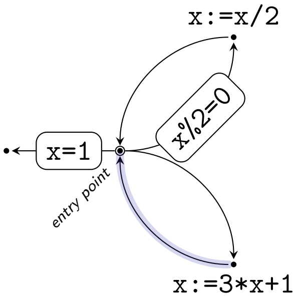
on a control flow graph



the current value of x is 40

An Execution Trace

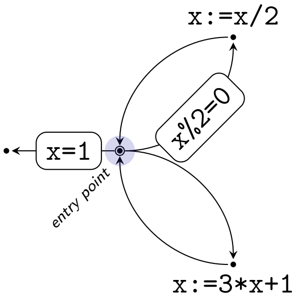
on a control flow graph



the current value of `x` is 40

An Execution Trace

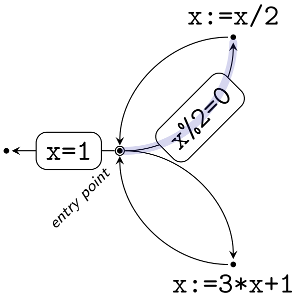
on a control flow graph



the current value of `x` is 40

An Execution Trace

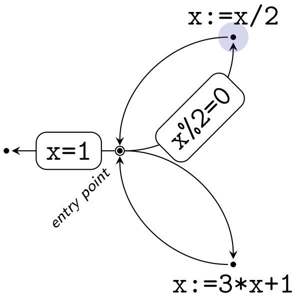
on a control flow graph



the current value of x is 40

An Execution Trace

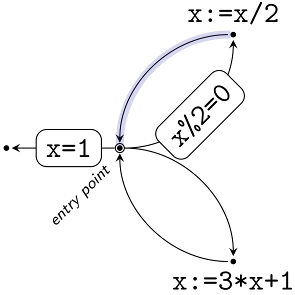
on a control flow graph



the current value of x is 20

An Execution Trace

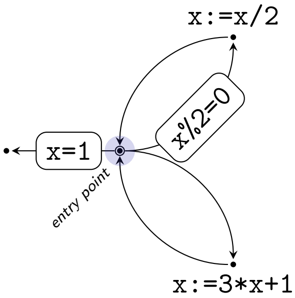
on a control flow graph



the current value of x is 20

An Execution Trace

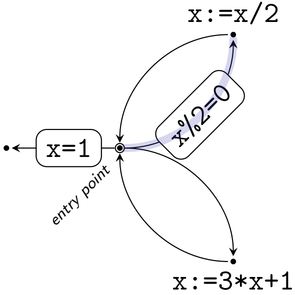
on a control flow graph



the current value of x is 20

An Execution Trace

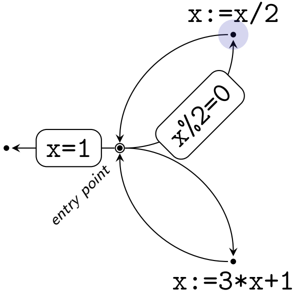
on a control flow graph



the current value of x is 20

An Execution Trace

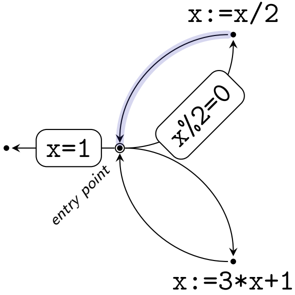
on a control flow graph



the current value of x is 10

An Execution Trace

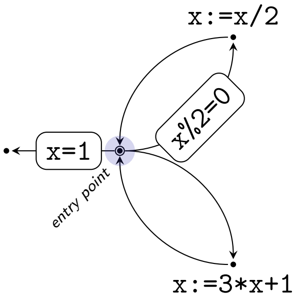
on a control flow graph



the current value of x is 10

An Execution Trace

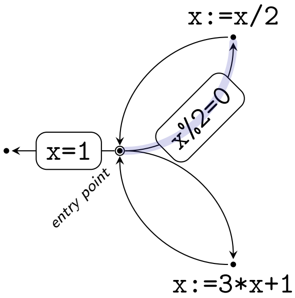
on a control flow graph



the current value of x is 10

An Execution Trace

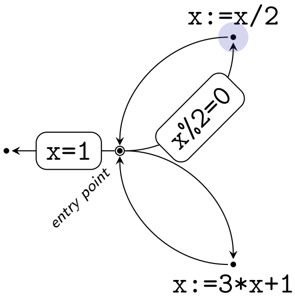
on a control flow graph



the current value of x is 10

An Execution Trace

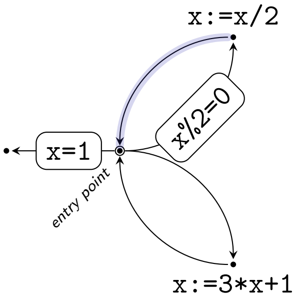
on a control flow graph



the current value of x is 5

An Execution Trace

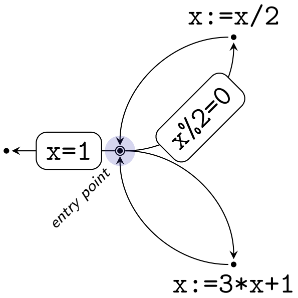
on a control flow graph



the current value of x is 5

An Execution Trace

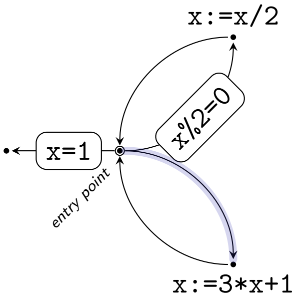
on a control flow graph



the current value of x is 5

An Execution Trace

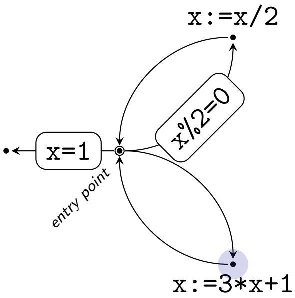
on a control flow graph



the current value of x is 5

An Execution Trace

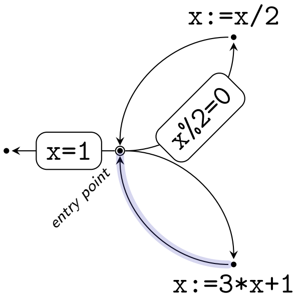
on a control flow graph



the current value of x is 16

An Execution Trace

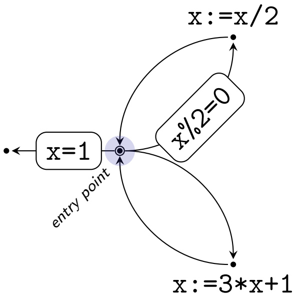
on a control flow graph



the current value of x is 16

An Execution Trace

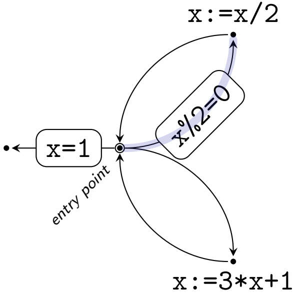
on a control flow graph



the current value of x is 16

An Execution Trace

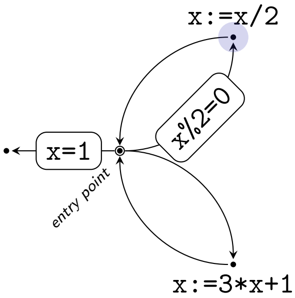
on a control flow graph



the current value of x is 16

An Execution Trace

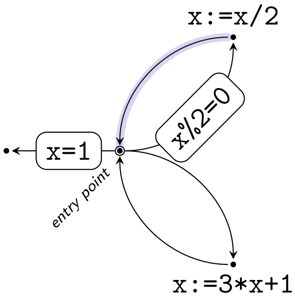
on a control flow graph



the current value of x is 8

An Execution Trace

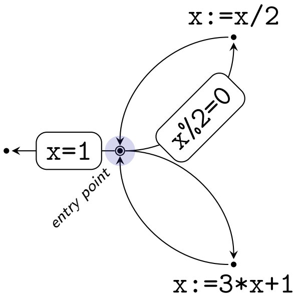
on a control flow graph



the current value of x is 8

An Execution Trace

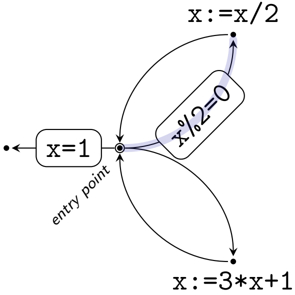
on a control flow graph



the current value of x is 8

An Execution Trace

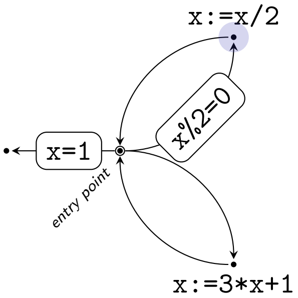
on a control flow graph



the current value of x is 8

An Execution Trace

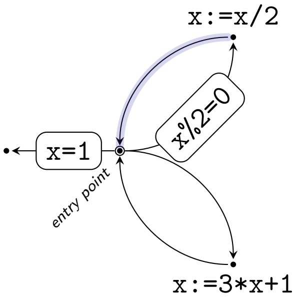
on a control flow graph



the current value of x is 4

An Execution Trace

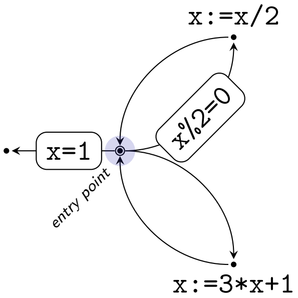
on a control flow graph



the current value of x is 4

An Execution Trace

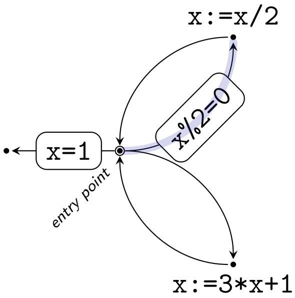
on a control flow graph



the current value of x is 4

An Execution Trace

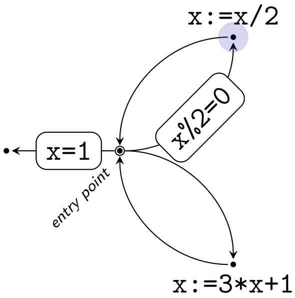
on a control flow graph



the current value of x is 4

An Execution Trace

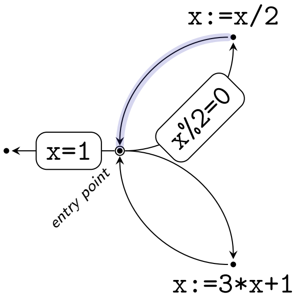
on a control flow graph



the current value of $\texttt{x}$ is 2

An Execution Trace

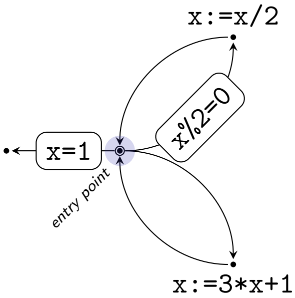
on a control flow graph



the current value of x is 2

An Execution Trace

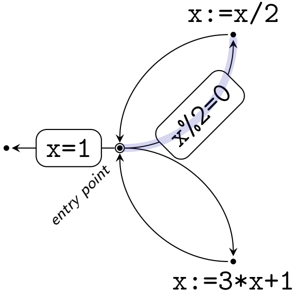
on a control flow graph



the current value of x is 2

An Execution Trace

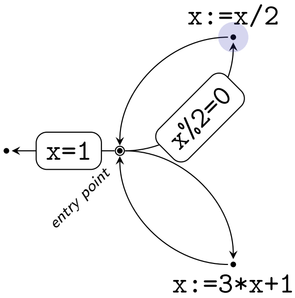
on a control flow graph



the current value of x is 2

An Execution Trace

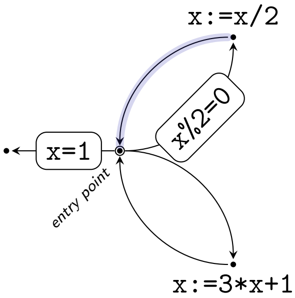
on a control flow graph



the current value of x is 1

An Execution Trace

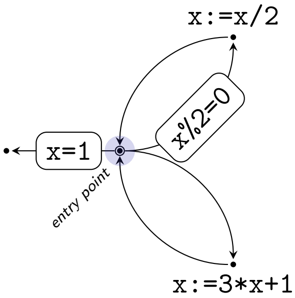
on a control flow graph



the current value of x is 1

An Execution Trace

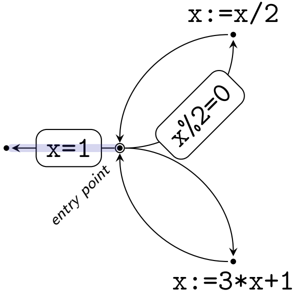
on a control flow graph



the current value of x is 1

An Execution Trace

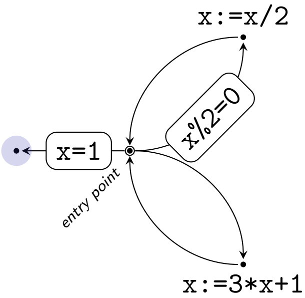
on a control flow graph



the current value of x is 1

An Execution Trace

on a control flow graph



the current value of x is 1

Precubical sets

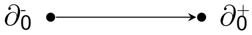
higher dimensional graphs

Precubical sets

higher dimensional graphs

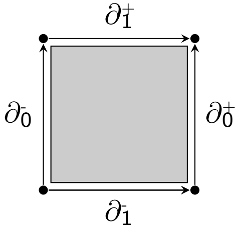
Precubical sets

higher dimensional graphs



Precubical sets

higher dimensional graphs



Precubical sets

higher dimensional graphs

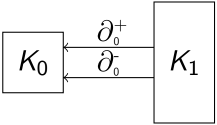
Precubical sets

higher dimensional graphs

K_0

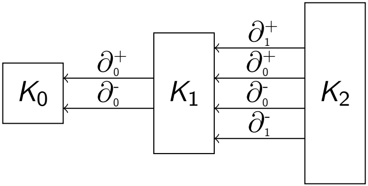
Precubical sets

higher dimensional graphs



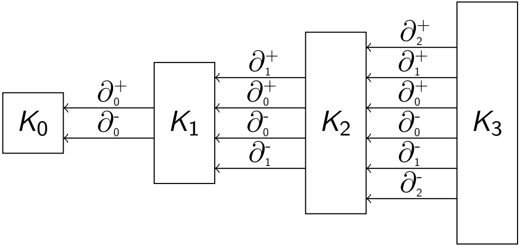
Precubical sets

higher dimensional graphs



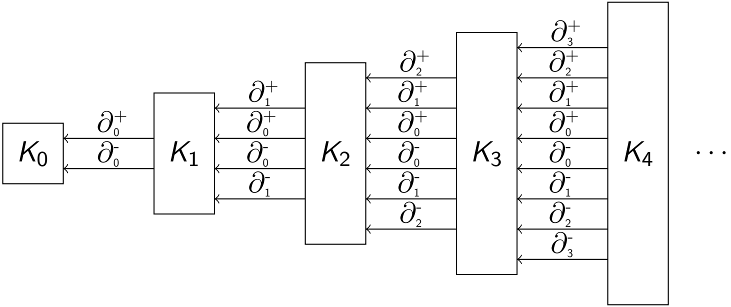
Precubical sets

higher dimensional graphs



Precubical sets

higher dimensional graphs



Tensor product

of precubical sets

Given precubical sets K and K' of dimension p and q, the set of d-cubes for $0 \leqslant d \leqslant p+q$

$$(K \otimes K')_d = \bigsqcup_{i+j=d} K_i \times K'_j$$

For $x \otimes y \in K_i \times K'_j$ with $i+j=d$ the k^{th} face map, with $0 \leqslant k < d$, is given by

$$\partial_k^{\pm}(x \otimes y) = \begin{cases} \partial_k^{\pm}(x) \otimes y & \text{if } 0 \leqslant k < i \\ x \otimes \partial_{k-p}^{\pm} y) & \text{if } i \leqslant k < d \end{cases}$$

A Toy Language

Synchronization: the W(_) instruction

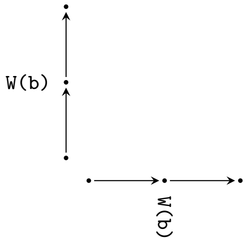
```
sync:  1 b
```

```
proc:  p = W(b)
```

```
init:  2p
```

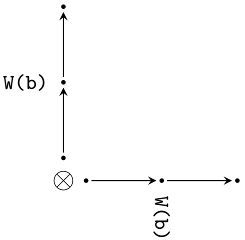
Tensor product

of control flow graphs



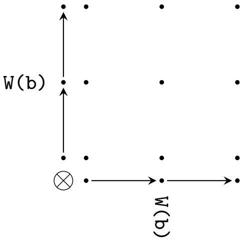
Tensor product

of control flow graphs



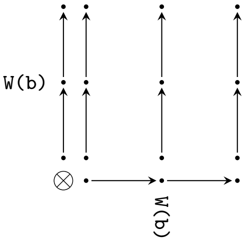
Tensor product

of control flow graphs



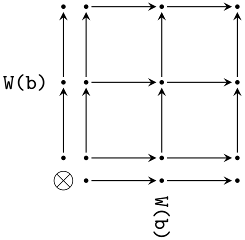
Tensor product

of control flow graphs



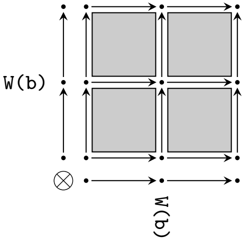
Tensor product

of control flow graphs



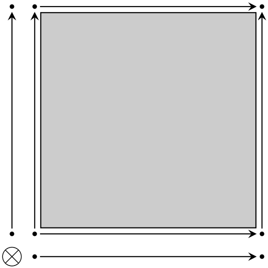
Tensor product

of control flow graphs



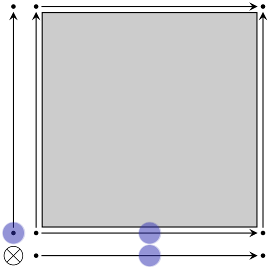
Discrete paths

are "continuous"



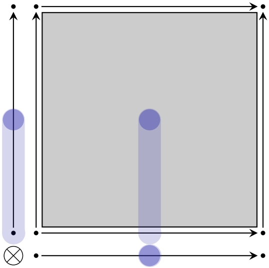
Discrete paths

are “continuous”



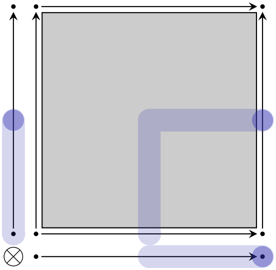
Discrete paths

are "continuous"



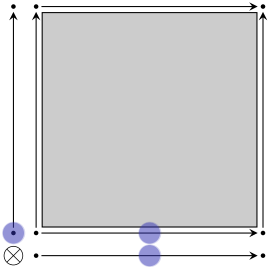
Discrete paths

are “continuous”



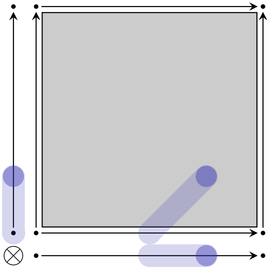
Discrete paths

are "continuous"



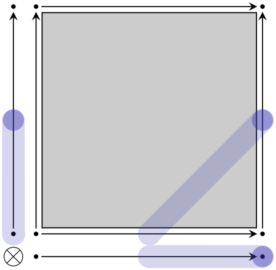
Discrete paths

are “continuous”



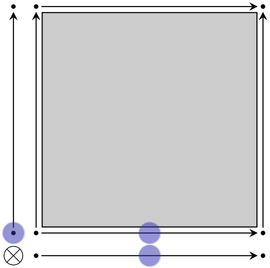
Discrete paths

are “continuous”



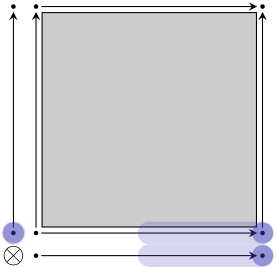
Discrete paths

are “continuous”



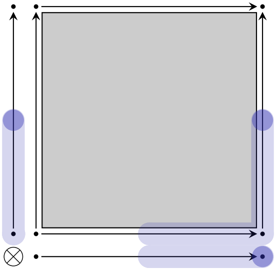
Discrete paths

are "continuous"



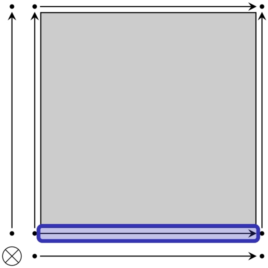
Discrete paths

are "continuous"



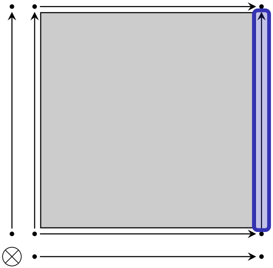
Discrete paths

are "continuous"



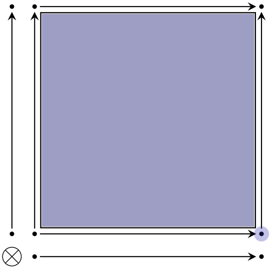
Discrete paths

are "continuous"



Discrete paths

are “continuous”



Discrete path on a model of dimension N

A sequence of points $p_0, \ldots, p_K$ s.t. for all $k \in \{1, \ldots, K\}$ one has

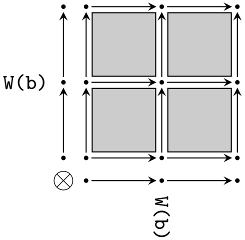
$$\text{for all } n \in \{1, \ldots, N\} \ \partial^+ p_n(k-1) = p_n(k) \text{ or } p_n(k) = p_n(k-1)$$

or

$$\text{for all } n \in \{1, \ldots, N\} \ p_n(k-1) = \partial^- p_n(k) \text{ or } p_n(k) = p_n(k-1)$$

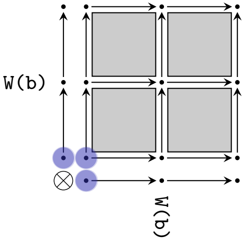
Concurrent execution trace

sync: 1 b



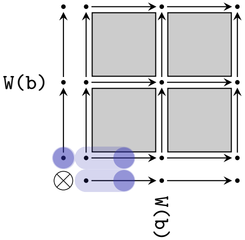
Concurrent execution trace

`sync:  1 b`



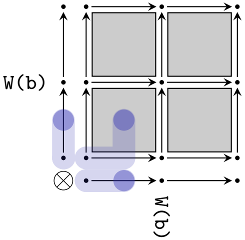
Concurrent execution trace

`sync: 1 b`



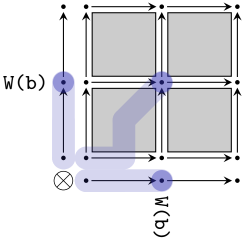
Concurrent execution trace

`sync:  1 b`



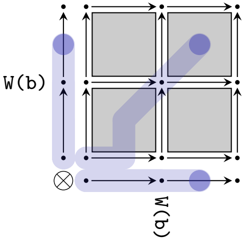
Concurrent execution trace

`sync:  1 b`



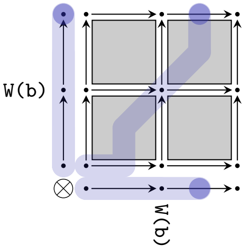
Concurrent execution trace

`sync: 1 b`



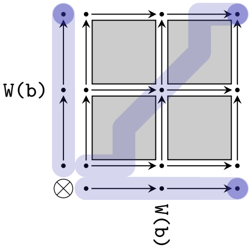
Concurrent execution trace

`sync:  1 b`



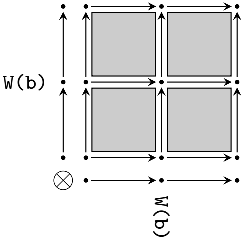
Concurrent execution trace

`sync: 1 b`



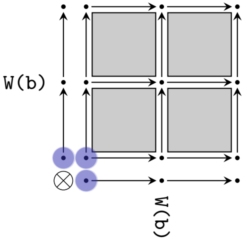
Not admissible concurrent execution trace

```
sync:  1 b
```



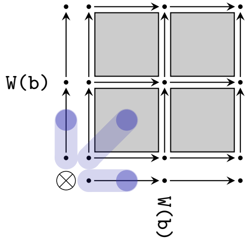
Not admissible concurrent execution trace

sync: 1 b



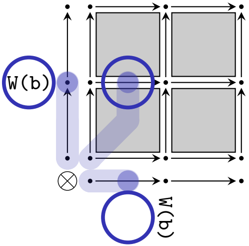
Not admissible concurrent execution trace

sync: 1 b



Not admissible concurrent execution trace

sync: 1 b



Forbidden points

due to synchronization

Each point $p = (p_1, \ldots, p_d)$ such that

$$0 \quad < \quad \mathrm{card}\Big\{ k \in \{1, \ldots, d\} \;\Big|\; \mathrm{label}(p_k) = \mathtt{W(b)} \Big\} \quad \leqslant \quad \mathrm{arity}(\mathtt{b})$$

is forbidden.

A Toy Language

conflicting assignments

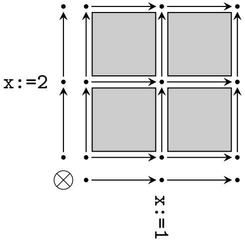
```
var:  x = 0
```

```
proc:  p = (x := 1)
proc:  q = (x := 2)
```

```
init:  p q
```

Not admissible execution trace

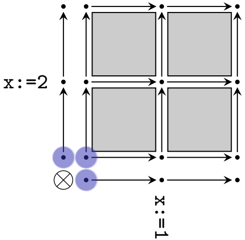
due to race condition



the value of $\mathtt{x}$ is 0

Not admissible execution trace

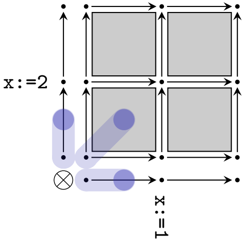
due to race condition



the value of $\mathtt{x}$ is 0

Not admissible execution trace

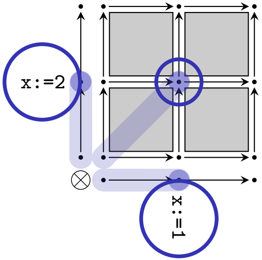
due to race condition



the value of $\mathtt{x}$ is 0

Not admissible execution trace

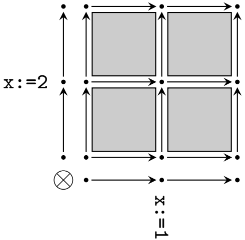
due to race condition



the value of x is ?

Admissible execution trace

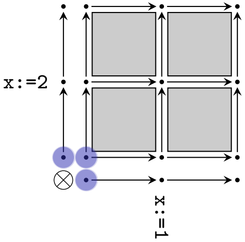
that however meets a forbidden point



the value of $\mathtt{x}$ is 0

Admissible execution trace

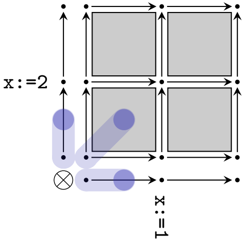
that however meets a forbidden point



the value of x is 0

Admissible execution trace

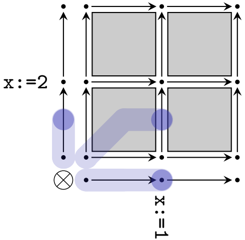
that however meets a forbidden point



the value of x is 0

Admissible execution trace

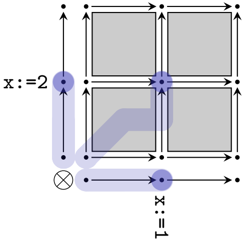
that however meets a forbidden point



the value of x is 1

Admissible execution trace

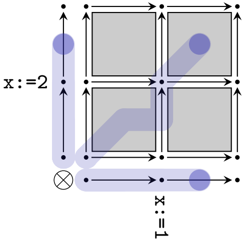
that however meets a forbidden point



the value of x is 2

Admissible execution trace

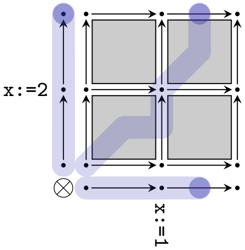
that however meets a forbidden point



the value of $\mathtt{x}$ is 2

Admissible execution trace

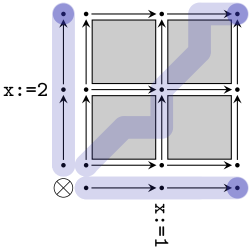
that however meets a forbidden point



the value of $\mathtt{x}$ is 2

Admissible execution trace

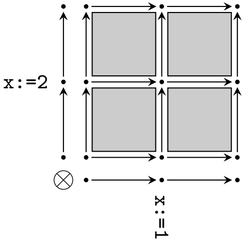
that however meets a forbidden point



the value of $\mathtt{x}$ is 2

Admissible execution trace

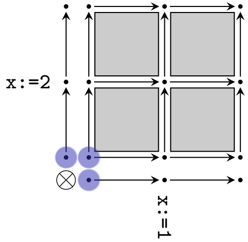
avoiding forbidden points



the value of $\mathtt{x}$ is 0

Admissible execution trace

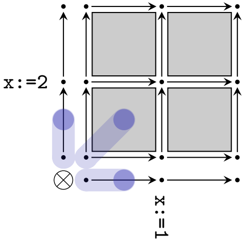
avoiding forbidden points



the value of $\texttt{x}$ is 0

Admissible execution trace

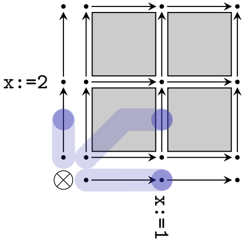
avoiding forbidden points



the value of x is 0

Admissible execution trace

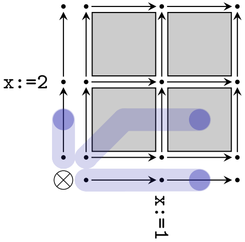
avoiding forbidden points



the value of x is 1

Admissible execution trace

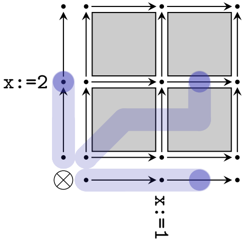
avoiding forbidden points



the value of x is 1

Admissible execution trace

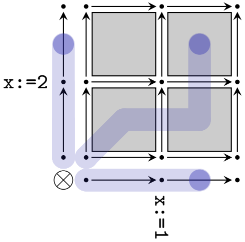
avoiding forbidden points



the value of $\texttt{x}$ is 2

Admissible execution trace

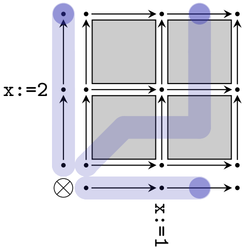
avoiding forbidden points



the value of x is 2

Admissible execution trace

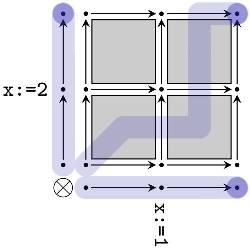
avoiding forbidden points



the value of $\mathtt{x}$ is 2

Admissible execution trace

avoiding forbidden points



the value of $\mathtt{x}$ is 2

Forbidden points

due to race conditions

A point $p = (p_1, \ldots, p_d)$ is a race condition when there exist $i \neq j$ such that
- both $\lambda_i(p_i)$ and $\lambda_j(p_j)$ are assignments trying to alter the same variable or
- $\lambda_i(p_i)$ tries to alter a free variable of $\lambda_j(p_j)$ or $\lambda_j(\alpha)$ for some arrow α such that $\partial^{-}\alpha = p_j$.

In that case the point p is forbidden.

The replacement property

for admissible execution traces

Replacement

Any admissible execution trace that meets a race condition is "equivalent" to an admissible execution trace which avoids all of them.

A Toy Language

Desynchronization: the P(_) and V(_) instructions

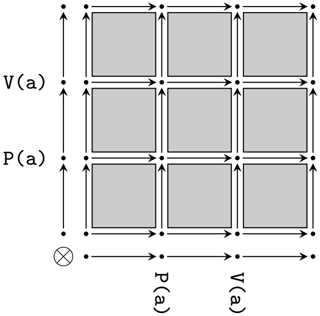
```
sem:  1 a
```

```
proc:  p = P(a);V(a)
```

```
init:  2p
```

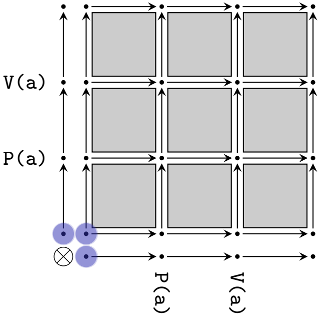
Admissible concurrent execution trace

`sem: 1 a`



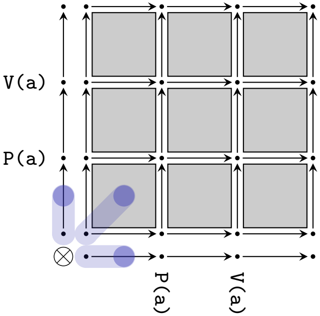
Admissible concurrent execution trace

`sem: 1 a`



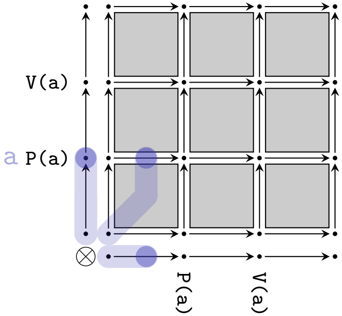
Admissible concurrent execution trace

sem: 1 a



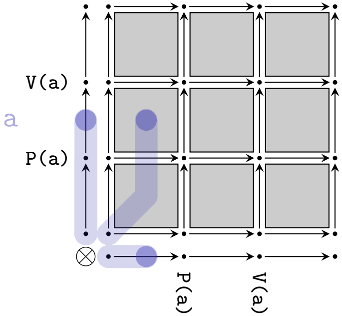
Admissible concurrent execution trace

`sem: 1 a`



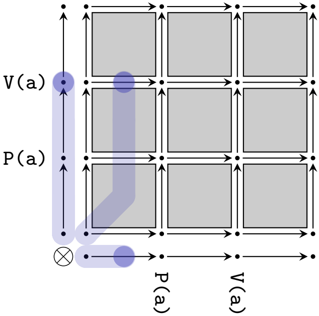
Admissible concurrent execution trace

`sem:  1 a`



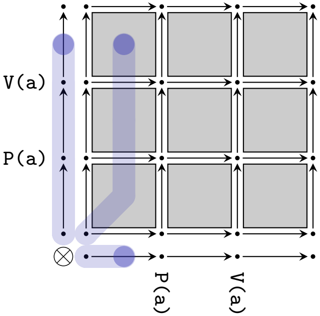
Admissible concurrent execution trace

`sem:  1 a`



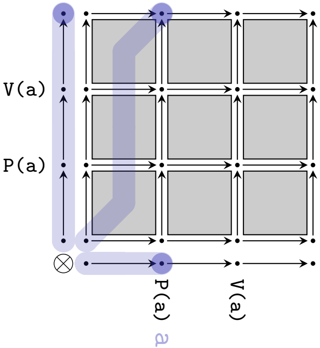
Admissible concurrent execution trace

sem: 1 a



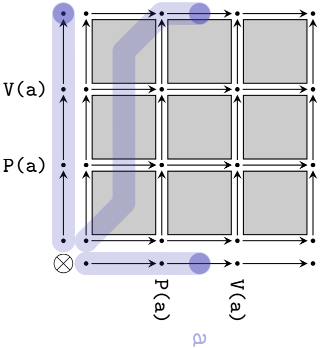
Admissible concurrent execution trace

sem: 1 a



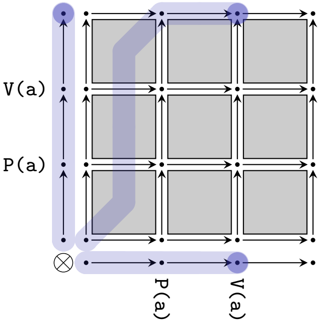
Admissible concurrent execution trace

sem: 1 a



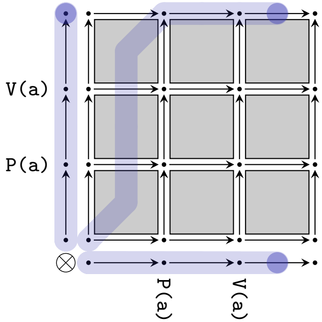
Admissible concurrent execution trace

`sem: 1 a`



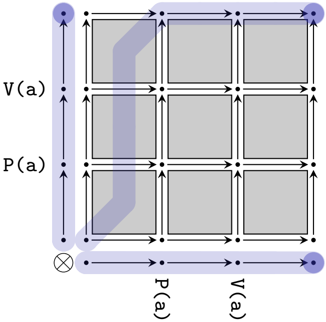
Admissible concurrent execution trace

`sem: 1 a`



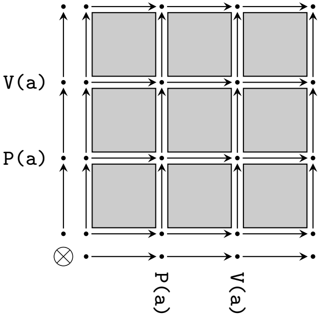
Admissible concurrent execution trace

`sem: 1 a`



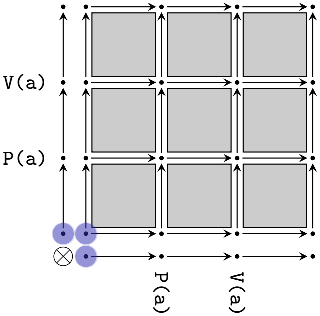
Not admissible concurrent execution trace

`sem: 1 a`



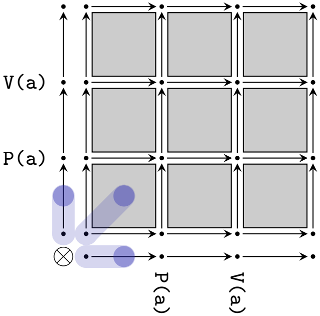
Not admissible concurrent execution trace

`sem:  1 a`



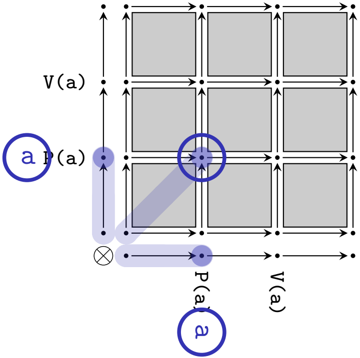
Not admissible concurrent execution trace

sem: 1 a



Not admissible concurrent execution trace

sem: 1 a



The potential functions

of processes and programs

The potential functions

of processes and programs

A process π is conservative when for all paths and all semaphores s, the amount of tokens of type s held by the process at the end of the execution trace only depends on its arrival point.

The potential functions

of processes and programs

A process π is conservative when for all paths and all semaphores s, the amount of tokens of type s held by the process at the end of the execution trace only depends on its arrival point. In that case the process π comes with a potential function F_π

$$F_\pi : \{\text{semaphores}\} \times \{\text{points}\} \to \mathbb{N}$$

The potential functions

of processes and programs

A process π is conservative when for all paths and all semaphores s, the amount of tokens of type s held by the process at the end of the execution trace only depends on its arrival point. In that case the process π comes with a potential function F_π

$$F_\pi : \{\text{semaphores}\} \times \{\text{points}\} \to \mathbb{N}$$

A program Π is conservative when so are its processes $\pi_1, \ldots, \pi_d$

The potential functions

of processes and programs

A process π is conservative when for all paths and all semaphores s, the amount of tokens of type s held by the process at the end of the execution trace only depends on its arrival point. In that case the process π comes with a potential function F_π

$$F_\pi : \{\text{semaphores}\} \times \{\text{points}\} \to \mathbb{N}$$

A program Π is conservative when so are its processes $\pi_1, \ldots, \pi_d$ and its potential function is given by

$$F_\Pi(s, (p_1, \ldots, p_d)) = \sum_{k=1}^{d} F_{\pi_k}(s, p_k)$$

The potential functions

of processes and programs

A process π is conservative when for all paths and all semaphores s, the amount of tokens of type s held by the process at the end of the execution trace only depends on its arrival point. In that case the process π comes with a potential function F_π

$$F_\pi : \{\text{semaphores}\} \times \{\text{points}\} \to \mathbb{N}$$

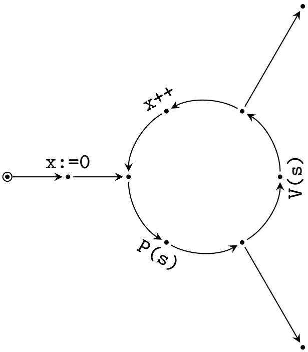
A program Π is conservative when so are its processes $\pi_1, \ldots, \pi_d$ and its potential function is given by

$$F_\Pi(s, (p_1, \ldots, p_d)) = \sum_{k=1}^{d} F_{\pi_k}(s, p_k)$$

If $F_\Pi(s, p) > \text{arity}(s)$ for some semaphore s, then p is forbidden.

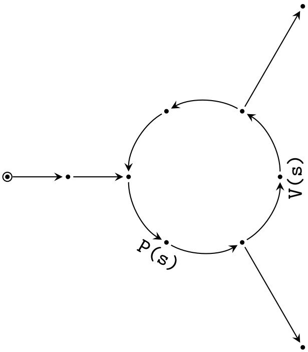
Conservative process

example



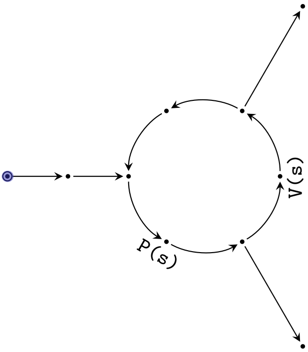
Conservative process

example



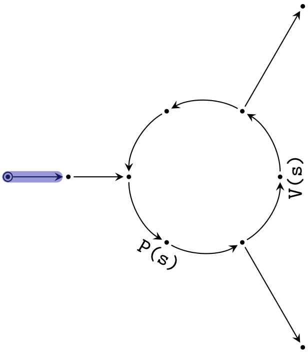
Conservative process

example



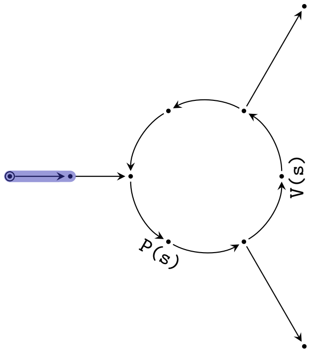
Conservative process

example



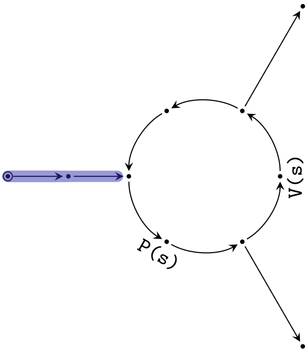
Conservative process

example



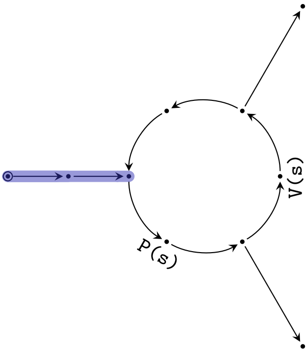
Conservative process

example



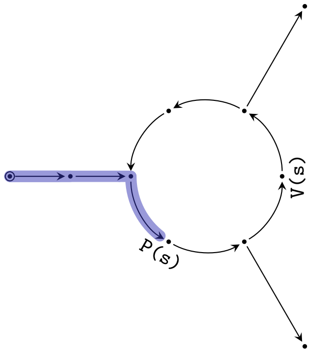
Conservative process

example



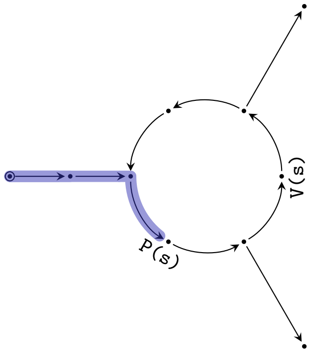
Conservative process

example



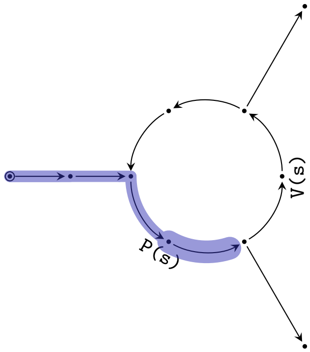
Conservative process

example



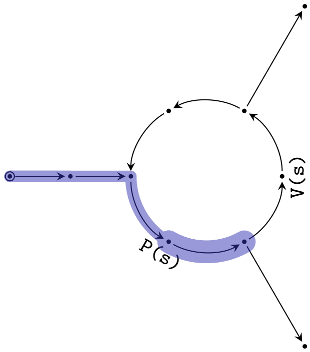
Conservative process

example



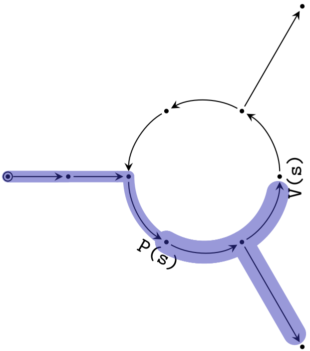
Conservative process

example



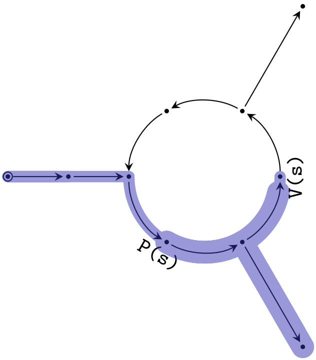
Conservative process

example



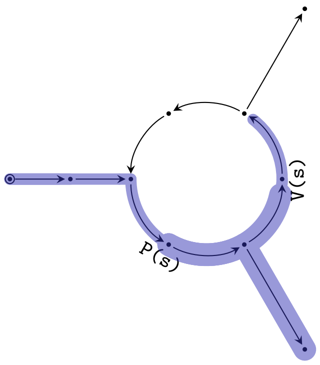
Conservative process

example



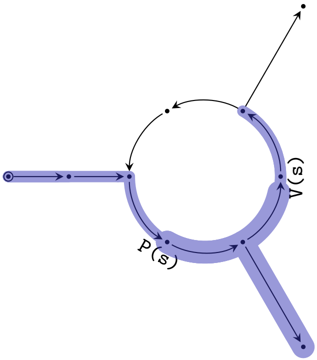
Conservative process

example



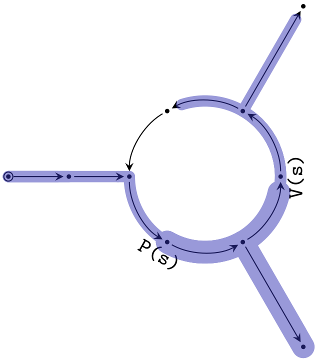
Conservative process

example



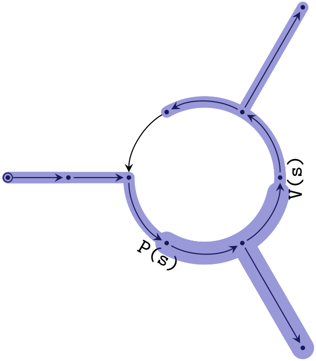
Conservative process

example



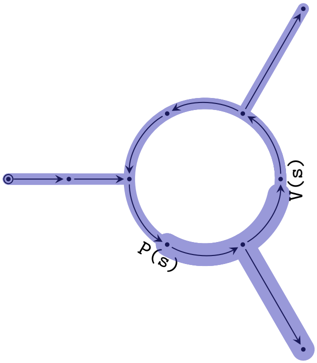
Conservative process

example



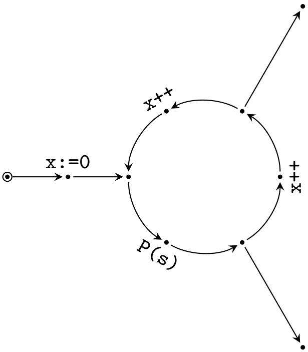
Conservative process

example



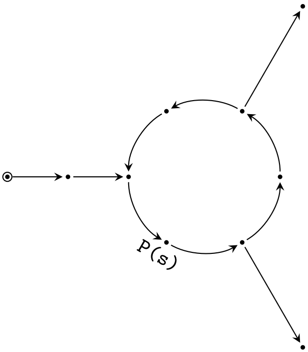
Not conservative process

example



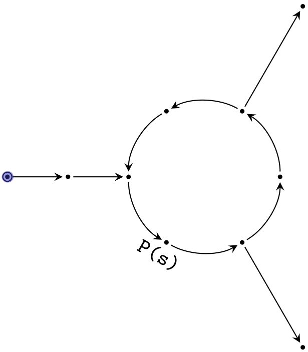
Not conservative process

example



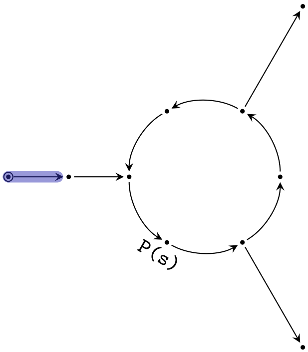
Not conservative process

example



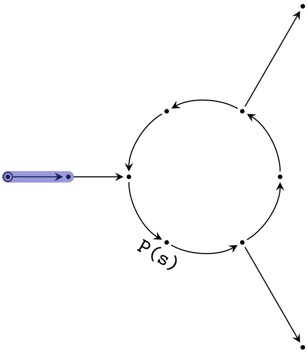
Not conservative process

example



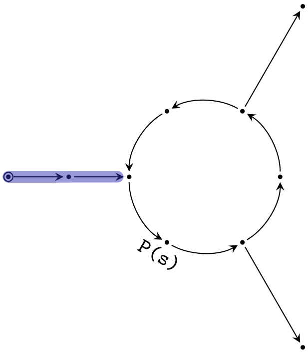
Not conservative process

example



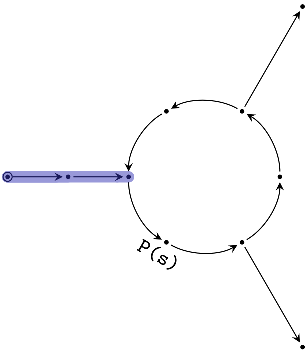
Not conservative process

example



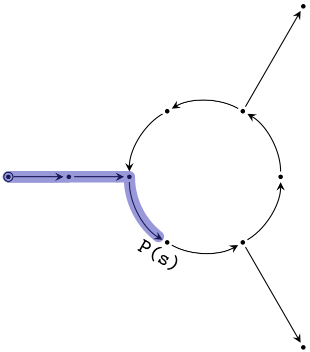
Not conservative process

example



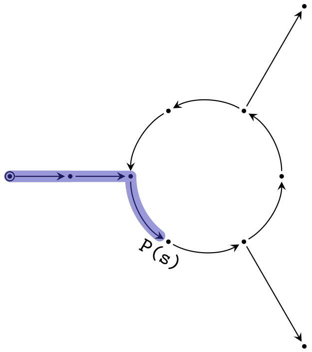
Not conservative process

example



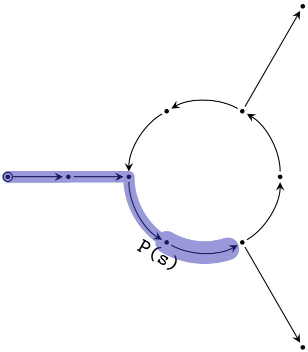
Not conservative process

example



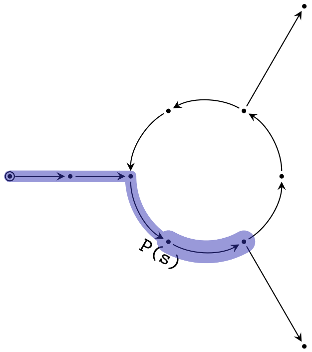
Not conservative process

example



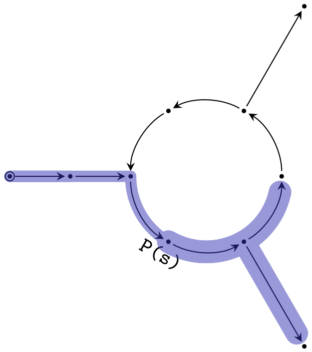
Not conservative process

example



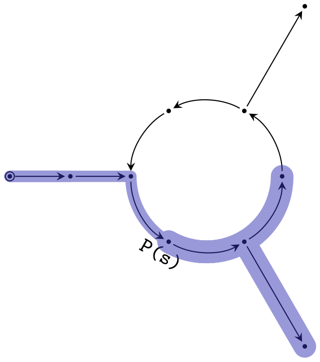
Not conservative process

example



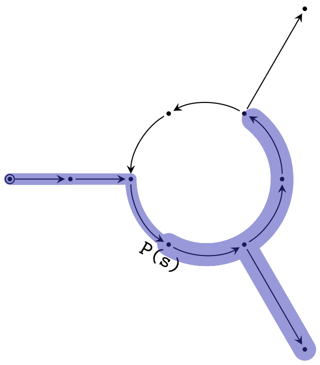
Not conservative process

example



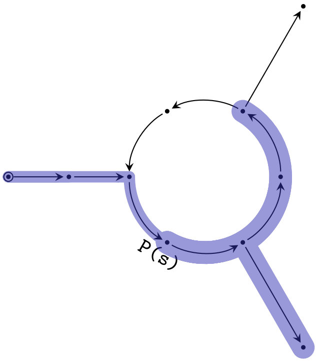
Not conservative process

example



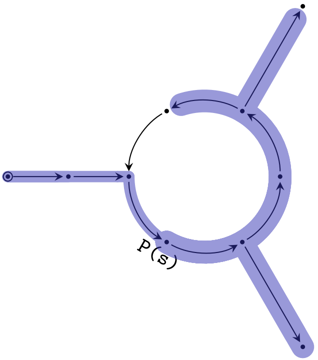
Not conservative process

example



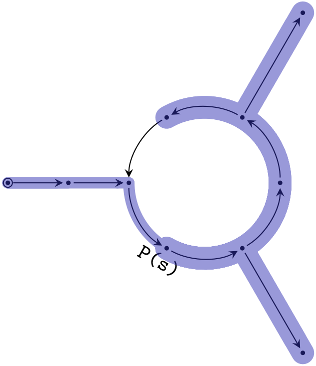
Not conservative process

example



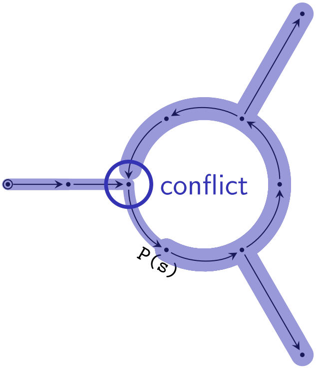
Not conservative process

example



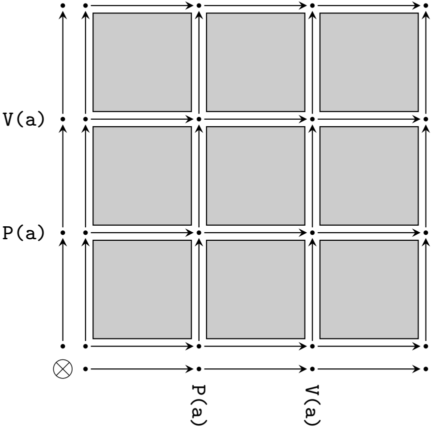
Not conservative process

example



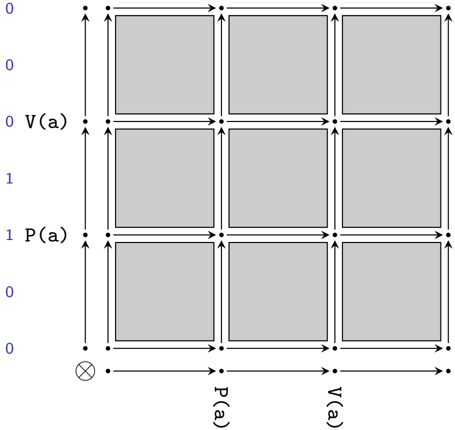
Discrete model

`sem: 1 a`



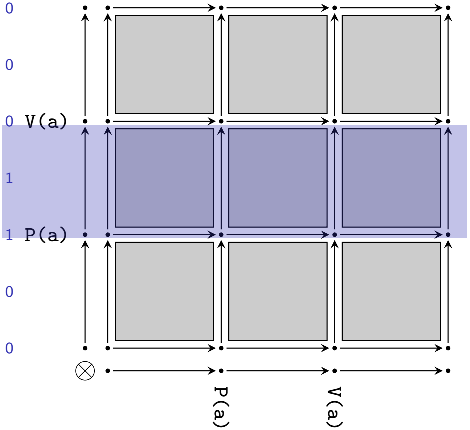
Discrete model

sem: 1 a



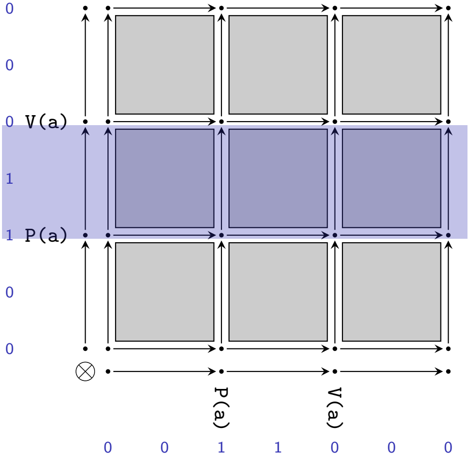
Discrete model

`sem: 1 a`



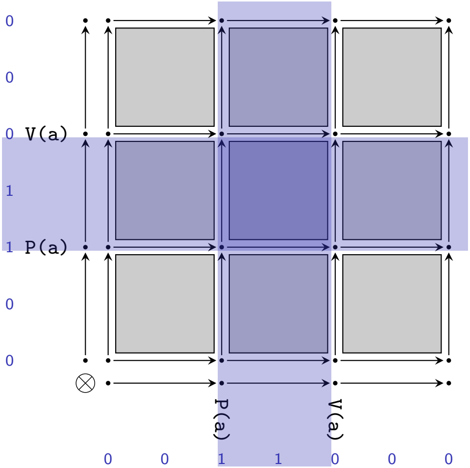
Discrete model

`sem:  1 a`



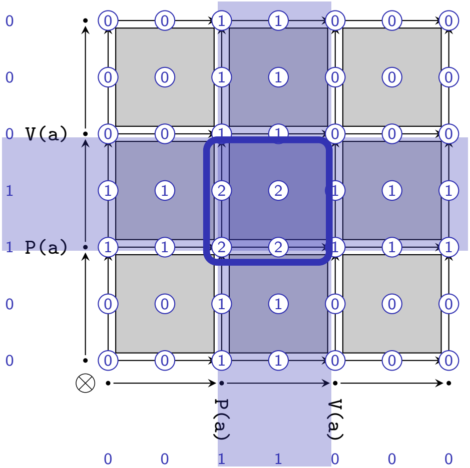
Discrete model

`sem:  1 a`



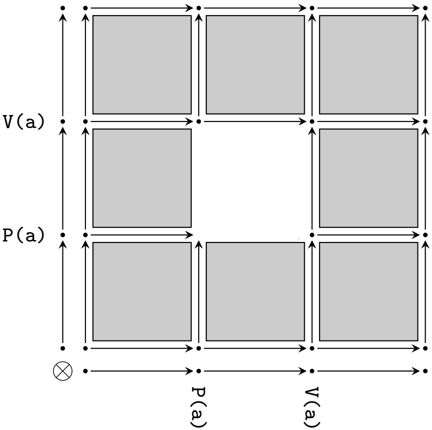
Discrete model

sem: 1 a



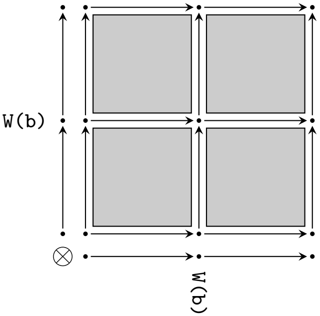
Discrete model

sem: 1 a



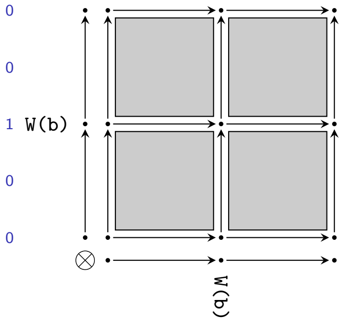
Discrete Model

sync: 1 b



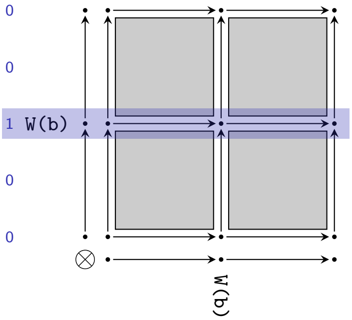
Discrete Model

sync: 1 b



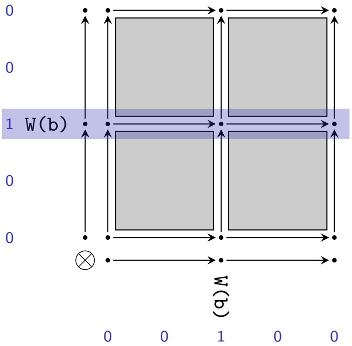
Discrete Model

sync: 1 b



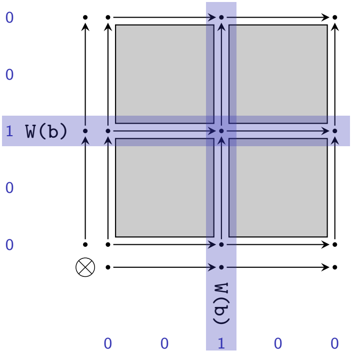
Discrete Model

`sync:  1 b`



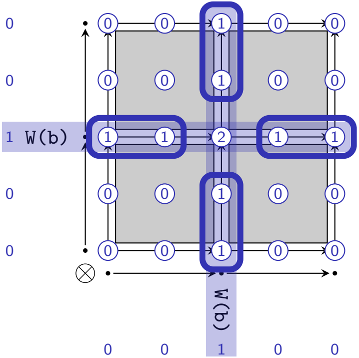
Discrete Model

`sync: 1 b`



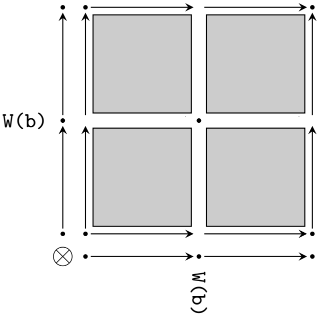
Discrete Model

`sync:  1 b`



Discrete Model

sync: 1 b



Locally ordered spaces

Directed atlas $\mathcal{U}$

Locally ordered spaces

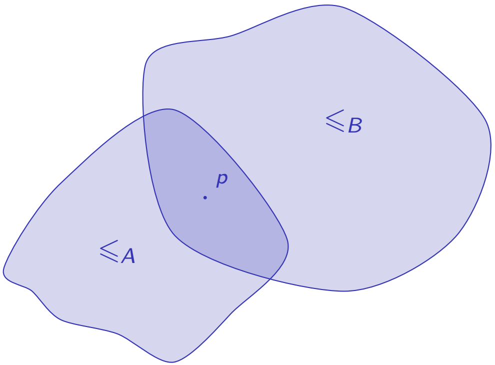
Directed atlas $\mathcal{U}$

For all points p,

Locally ordered spaces

Directed atlas $\mathcal{U}$

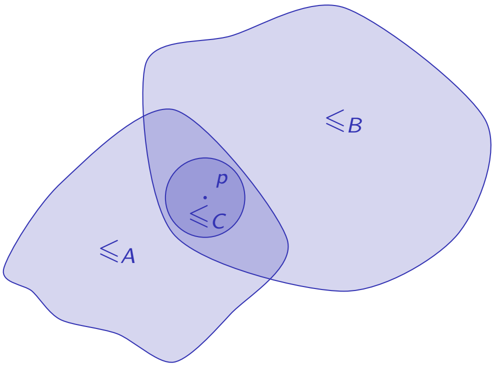
For all points p, for all directed neighborhoods A and B of p,



Locally ordered spaces

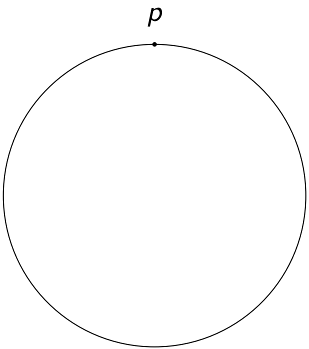
Directed atlas $\mathcal{U}$

For all points p, for all directed neighborhoods A and B of p, there exists a directed neighborhood C of p such that $C \subseteq A \cap B$ and $\leqslant_A|_C = \leqslant_C = \leqslant_B|_C$.



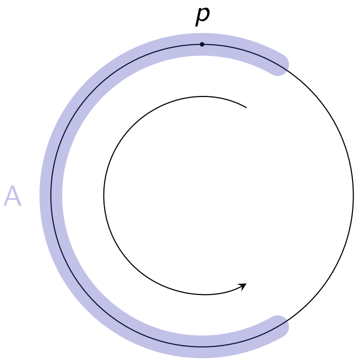
The directed circle

as a local pospace



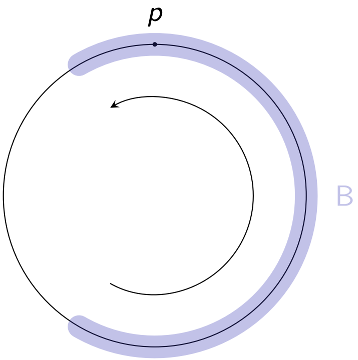
The directed circle

as a local pospace



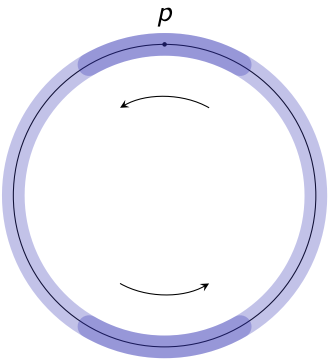
The directed circle

as a local pospace



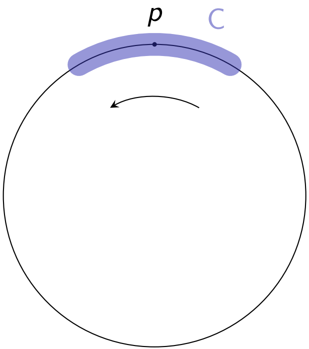
The directed circle

as a local pospace



The directed circle

as a local pospace



Directed geometric realization

Main property

$$\lceil \_ \rfloor : \{\text{Precubical sets}\} \to \{\text{Locally ordered spaces}\}$$

Directed geometric realization

Main property

$$\upharpoonleft\_\downharpoonright : \{\text{Precubical sets}\} \to \{\text{Locally ordered spaces}\}$$

$$U(\upharpoonleft K \downharpoonright) \quad = \quad \bigsqcup_{d \in \mathbb{N}} K_d \times]0,1[^d$$

Directed geometric realization

Main property

$$\upharpoonleft\_\downharpoonleft : \{\text{Precubical sets}\} \to \{\text{Locally ordered spaces}\}$$

$$U(\upharpoonleft K \downharpoonleft) \quad = \quad \bigsqcup_{d\in\mathbb{N}} K_d \times]0,1[^d$$

The main property

$$\upharpoonleft K^{(1)} \otimes \cdots \otimes K^{(n)} \downharpoonleft \quad \cong \quad \upharpoonleft K^{(1)} \downharpoonleft \times \cdots \times \upharpoonleft K^{(n)} \downharpoonleft$$

The continuous model

of a conservative program

Let $G^{(1)}, \ldots, G^{(n)}$ the control flow graphs of the program.

The continuous model

of a conservative program

Let $G^{(1)}, \dots, G^{(n)}$ the control flow graphs of the program.
For all $d \in \mathbb{N}$, the set F_d of forbidden points of dimension d.

The continuous model

of a conservative program

Let $G^{(1)}, \ldots, G^{(n)}$ the control flow graphs of the program.
For all $d \in \mathbb{N}$, the set F_d of forbidden points of dimension d.

$$K = G^{(1)} \otimes \cdots \otimes G^{(n)}$$

The continuous model

of a conservative program

Let $G^{(1)}, \ldots, G^{(n)}$ the control flow graphs of the program.
For all $d \in \mathbb{N}$, the set F_d of forbidden points of dimension d.

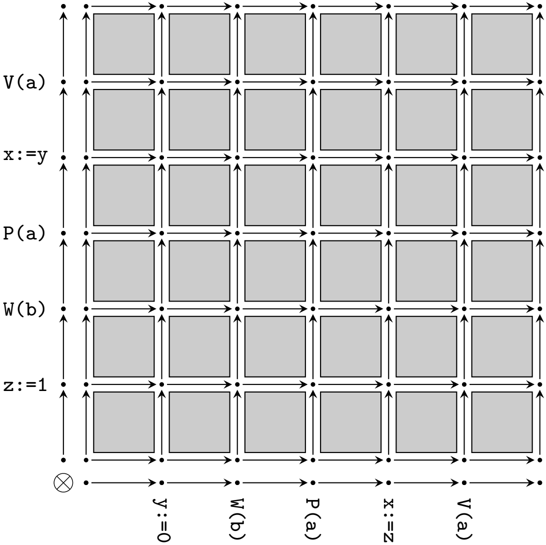
$$K = G^{(1)} \otimes \cdots \otimes G^{(n)}$$

The continuous model

$$\bigsqcup_{d \in \mathbb{N}} (K_d \setminus F_d) \times]0,1[^d$$

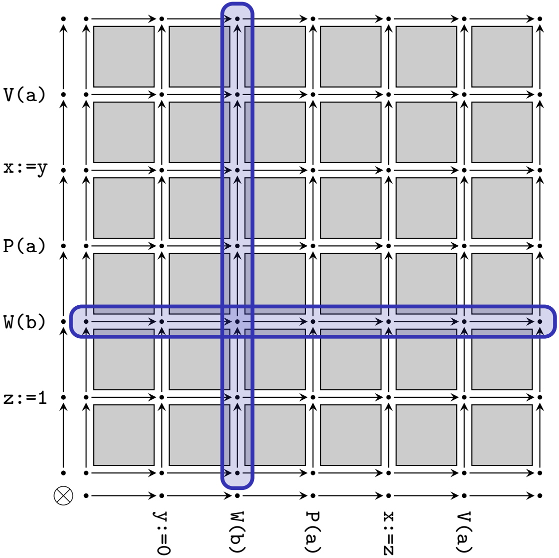
From discrete to continuous

sem: 1 a   sync: 1 b



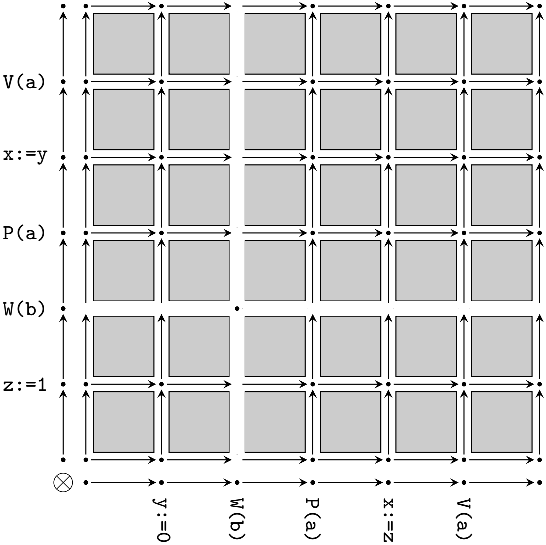
From discrete to continuous

sem: 1 a sync: 1 b



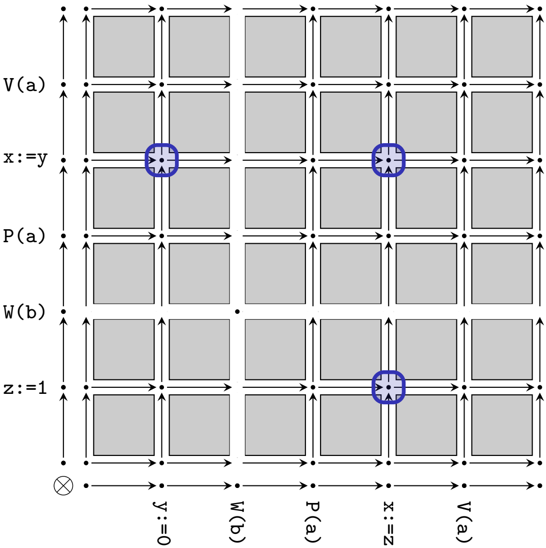
From discrete to continuous

sem: 1 a   sync: 1 b



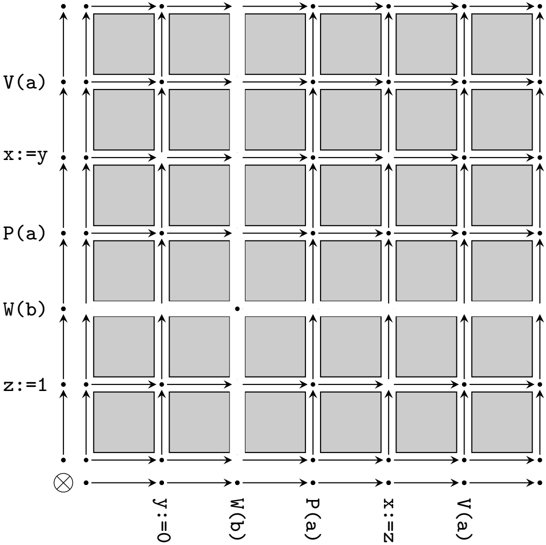
From discrete to continuous

sem: 1 a   sync: 1 b



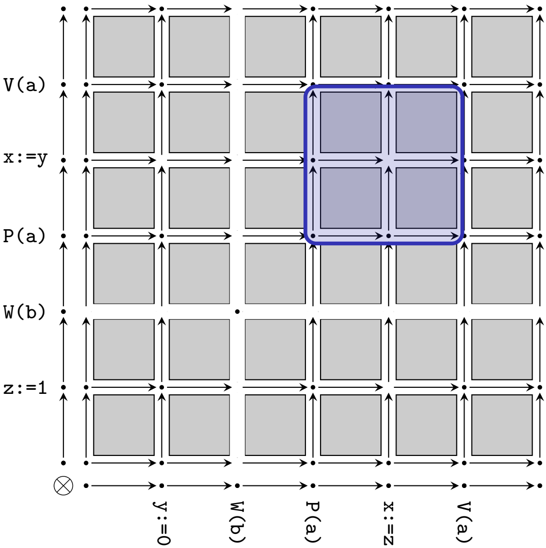
From discrete to continuous

`sem: 1 a` `sync: 1 b`



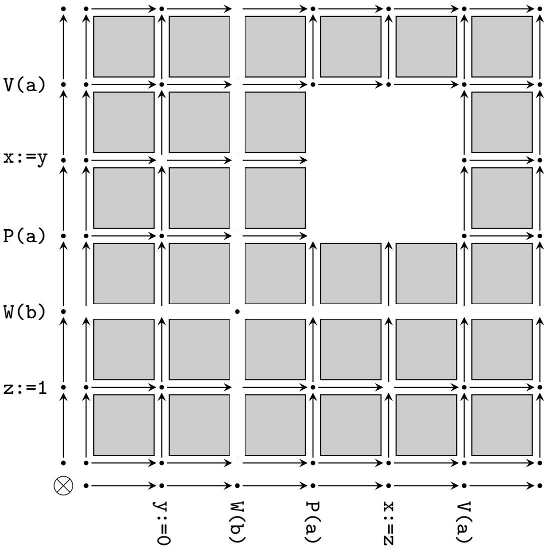
From discrete to continuous

sem: 1 a   sync: 1 b



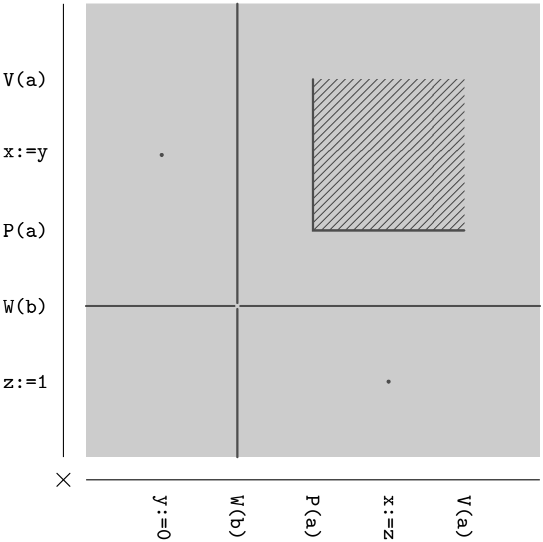
From discrete to continuous

sem: 1 a sync: 1 b



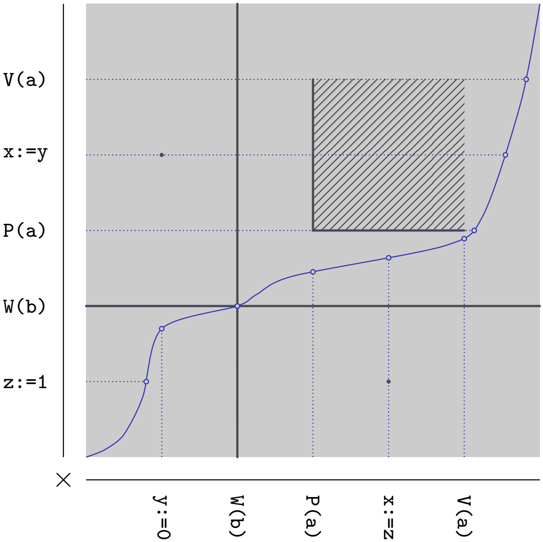
From discrete to continuous

`sem: 1 a        sync: 1 b`



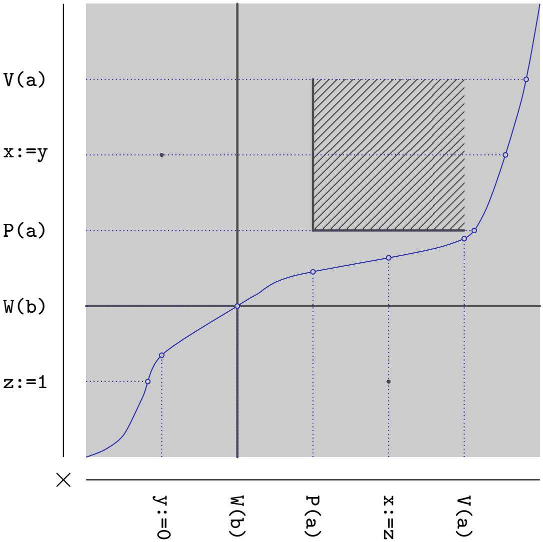
From discrete to continuous

sem: 1 a sync: 1 b



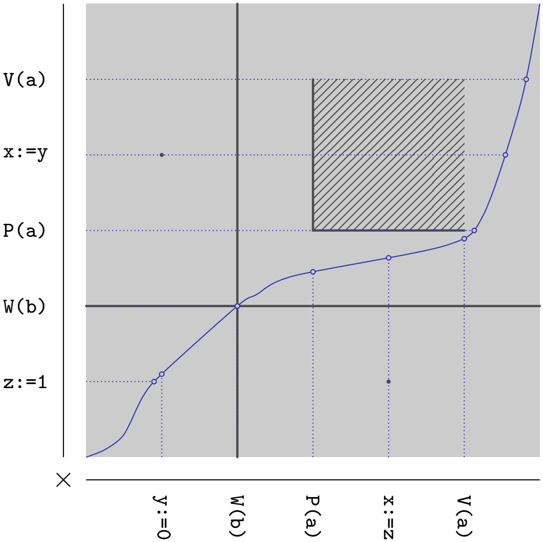
From discrete to continuous

sem: 1 a   sync: 1 b



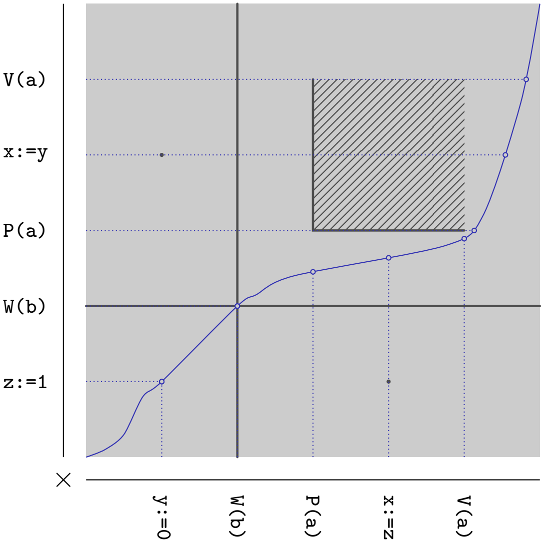
From discrete to continuous

sem: 1 a sync: 1 b



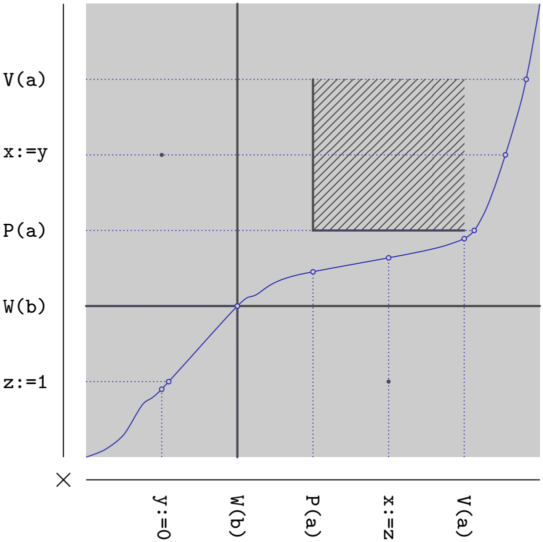
From discrete to continuous

sem: 1 a   sync: 1 b



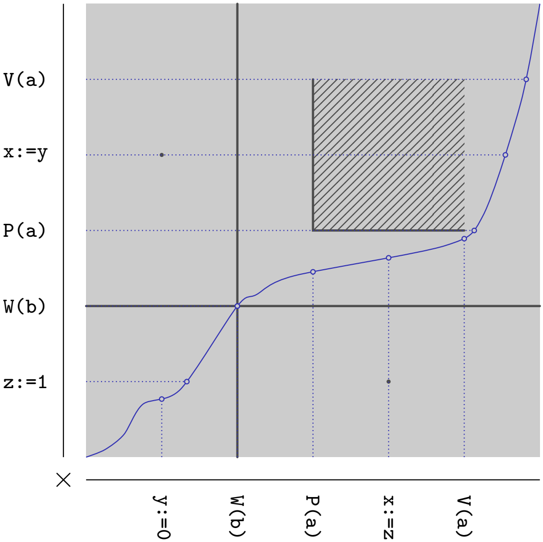
From discrete to continuous

sem: 1 a   sync: 1 b



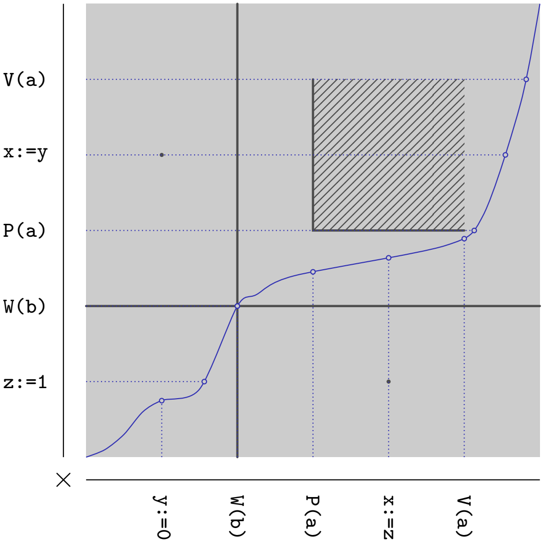
From discrete to continuous

sem: 1 a   sync: 1 b



From discrete to continuous

sem: 1 a   sync: 1 b



Weakly directed homotopy of directed paths

L. Fajstrup, É. Goubault, and M. Raussen (1998)

A weakly directed homotopy is a continuous map $h : [0, r] \times [0, q] \to X$ such that

Weakly directed homotopy of directed paths

L. Fajstrup, É. Goubault, and M. Raussen (1998)

A weakly directed homotopy is a continuous map $h : [0, r] \times [0, q] \to X$ such that

1) the mappings $h(0, -)$ and $h(r, -)$ are constant

Weakly directed homotopy of directed paths

L. Fajstrup, É. Goubault, and M. Raussen (1998)

A weakly directed homotopy is a continuous map $h : [0, r] \times [0, q] \to X$ such that

1) the mappings $h(0, -)$ and $h(r, -)$ are constant

2) the mappings $h(-, s)$ are directed paths

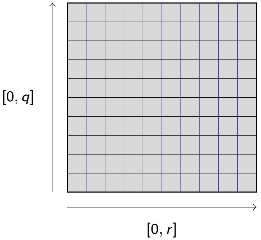
Weakly directed homotopy of directed paths

L. Fajstrup, É. Goubault, and M. Raussen (1998)

A weakly directed homotopy is a continuous map $h : [0, r] \times [0, q] \to X$ such that

1) the mappings $h(0, -)$ and $h(r, -)$ are constant

2) the mappings $h(-, s)$ are directed paths



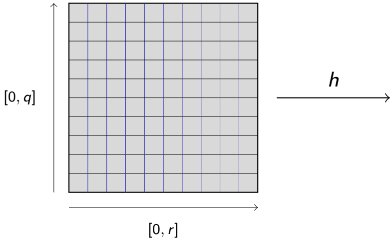
Weakly directed homotopy of directed paths

L. Fajstrup, É. Goubault, and M. Raussen (1998)

A weakly directed homotopy is a continuous map $h : [0, r] \times [0, q] \to X$ such that

1) the mappings $h(0, -)$ and $h(r, -)$ are constant

2) the mappings $h(-, s)$ are directed paths



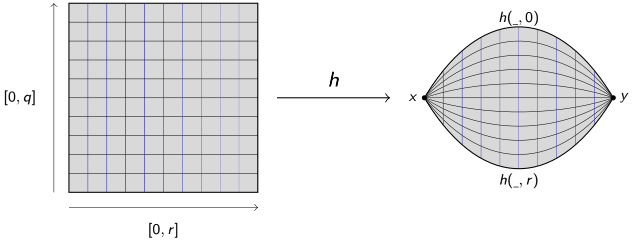
Weakly directed homotopy of directed paths

L. Fajstrup, É. Goubault, and M. Raussen (1998)

A weakly directed homotopy is a continuous map $h : [0, r] \times [0, q] \to X$ such that

1) the mappings $h(0, -)$ and $h(r, -)$ are constant

2) the mappings $h(-, s)$ are directed paths



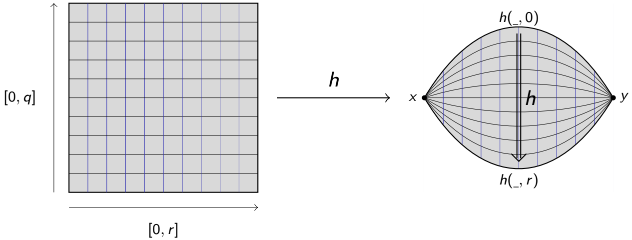
Weakly directed homotopy of directed paths

L. Fajstrup, É. Goubault, and M. Raussen (1998)

A weakly directed homotopy is a continuous map $h : [0, r] \times [0, q] \to X$ such that

1) the mappings $h(0, -)$ and $h(r, -)$ are constant

2) the mappings $h(-, s)$ are directed paths



Substantiating the continuous models

Main theorem

Adequacy

The “actions” of weakly dihomotopic directed paths are the same. A directed path is an execution trace iff it is weakly dihomotopic with an execution trace.

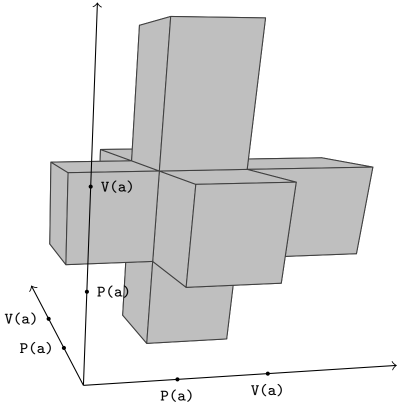
Tetrahemihexacron

a.k.a. 3D Swiss Cross

```
sem:  1 a
```

```
proc:
  p = P(a);V(a)
```

```
init:  3p
```



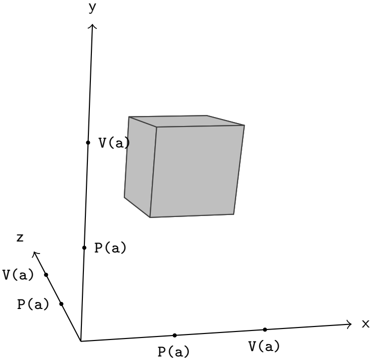
Floating cube

influence of arity

```
sem:  2 a
```

```
proc:
  p = P(a);V(a)
```

```
init:  3p
```



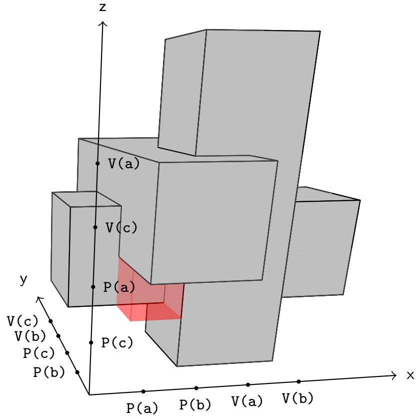
The dining philosophers

with its deadlock attractor

```
sem:  1 a b c
```

```
proc:
  x = P(a);P(b);V(a);V(b)
  y = P(b);P(c);V(b);V(c)
  z = P(c);P(a);V(c);V(a)
```

```
init:  x y z
```



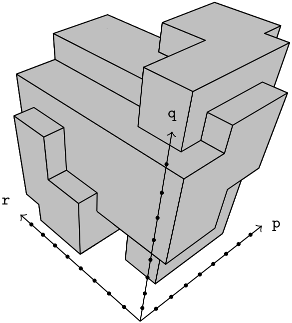
The Lipski algorithm

has no deadlock

```
sem:  1 x y z u v w
```

```
proc:
  p = P(x);P(y);P(z);V(x);P(w);V(z);V(y);V(w)
  q = P(u);P(v);P(x);V(u);P(z);V(v);V(x);V(z)
  r = P(y);P(w);V(y);P(u);V(w);P(v);V(u);V(v)
```

```
init:  p q r
```



Regions

over $G_1, \ldots, G_d$

A one dimensional block over G is a finite union of connected components of $\rceil G \lfloor$.

Regions

over $G_1, \ldots, G_d$

A one dimensional block over G is a finite union of connected components of $\rceil G \lfloor$.

A block of dimension $d \in \mathbb{N}$ over $G_1, \ldots, G_d$ is a Cartesian product of one dimensional blocks B_k over G_k for $k \in \{1, \ldots, d\}$.

Regions

over $G_1, \ldots, G_d$

A one dimensional block over G is a finite union of connected components of $\rceil G \lfloor$.

A block of dimension $d \in \mathbb{N}$ over $G_1, \ldots, G_d$ is a Cartesian product of one dimensional blocks B_k over G_k for $k \in \{1, \ldots, d\}$.

A region of dimension $d \in \mathbb{N}$ over $G_1, \ldots, G_d$ is a finite union of d-blocks over $G_1, \ldots, G_d$.

Regions

over $G_1, \ldots, G_d$

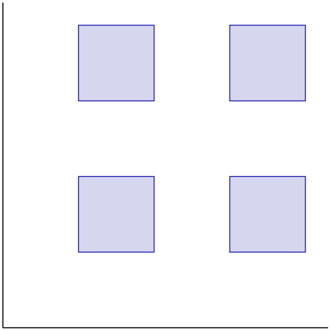
A one dimensional block over G is a finite union of connected components of $\rceil G \lfloor$.

A block of dimension $d \in \mathbb{N}$ over $G_1, \ldots, G_d$ is a Cartesian product of one dimensional blocks B_k over G_k for $k \in \{1, \ldots, d\}$.

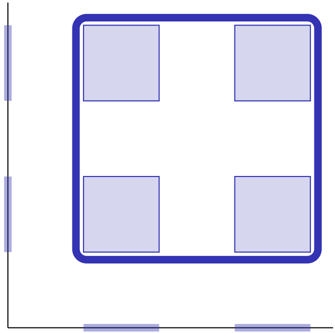
A region of dimension $d \in \mathbb{N}$ over $G_1, \ldots, G_d$ is a finite union of d-blocks over $G_1, \ldots, G_d$.

If X and Y are regions over $G_1, \ldots, G_d$ and $G'_1, \ldots, G'_{d'}$ then $X \times Y$ is a region over $G_1, \ldots, G_d, G'_1, \ldots, G'_{d'}$.

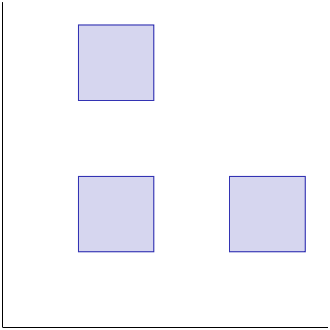
Maximal blocks



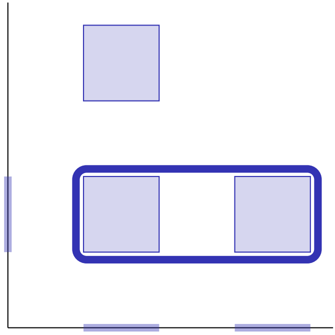
Maximal blocks



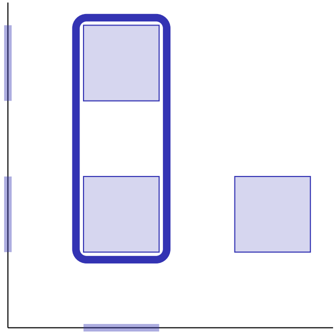
Maximal blocks



Maximal blocks



Maximal blocks



Main results

Maximal subblocks and Boolean structure

Main results

Maximal subblocks and Boolean structure

Maximal subblocks

$X \subseteq \upharpoonleft G_1 \downharpoonleft \times \cdots \times \upharpoonleft G_d \downharpoonleft$ is a region iff it has finitely many maximal subblocks.

Main results

Maximal subblocks and Boolean structure

Maximal subblocks

$X \subseteq \upharpoonleft G_1 \downharpoonright \times \cdots \times \upharpoonleft G_d \downharpoonright$ is a region iff it has finitely many maximal subblocks.

Boolean structure

The collection of regions over $G_1, \ldots, G_d$ form a Boolean subalgebra of the powerset of $\upharpoonleft G_1 \downharpoonright \times \cdots \times \upharpoonleft G_d \downharpoonright$.

Main results

Unique decomposition

Prime decomposition

Up to coordinates reordering, any region can be written as a Cartesian product of irreducible regions in a unique way. This is the prime decomposition of it.

Prime decomposition

Up to coordinates reordering, any region can be written as a Cartesian product of irreducible regions in a unique way. This is the prime decomposition of it.

Parallelization of code

The prime decomposition of the continuous model of some program provides a decomposition of the program as a parallel compound of "observationally independent" programs.

Main result

Effectiveness

An algorithm (Nicolas Ninin)

Let $M_1, \ldots, M_b$ be the maximal subblocks of $X^c = \upharpoonleft G_1 \downharpoonright \times \cdots \times \upharpoonleft G_d \downharpoonright \setminus X$. Let $\sim$ be the equivalence relation on $\{1, \ldots, d\}$ generated by $i \sim j$ when there exist $k \in \{1, \ldots, b\}$ such that

$$\mathrm{proj}_i(M_k) \neq \upharpoonleft G_i \downharpoonright \quad \text{and} \quad \mathrm{proj}_j(M_k) \neq \upharpoonleft G_j \downharpoonright$$

The prime decomposition of X is given by the $\sim$-equivalence classes.

Parallelizing a program

```
sem:  1 a b
sem:  2 c
```

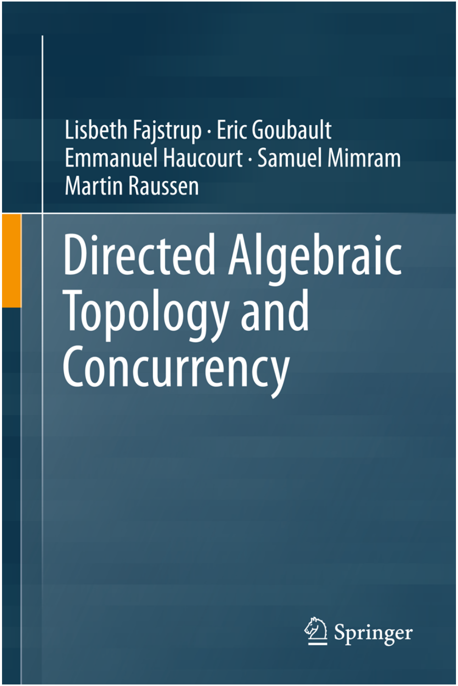
```
proc:
  p = P(a);P(c);V(c);V(a)

proc:
  q = P(b);P(c);V(c);V(b)
```

```
init:  2p 2q
```

Parallelizing a program

sem: 1 a b sem: 2 c	sem: 1 a b sem: 2 c
proc: p = P(a);P(c);V(c);V(a)	proc: q = P(b);P(c);V(c);V(b)
init: 2p	init: 2q


Lisbeth Fajstrup · Eric Goubault
Emmanuel Haucourt · Samuel Mimram
Martin Raussen
Directed Algebraic Topology and Concurrency
Springer