Toward the concept of backtracking computation*

Marija Kulaš
FernUniversität Hagen, Germany

Abstract

S_1:PP is a new mathematical definition of the Byrd's box model for pure Prolog, with:

- Forward and backward steps
- Modularity

---

*paper presented at the SOS'04 Workshop, 30. August 2004, London

Motivation: How to specify (and even prove) properties of computation?

What does it mean that a run-time test is

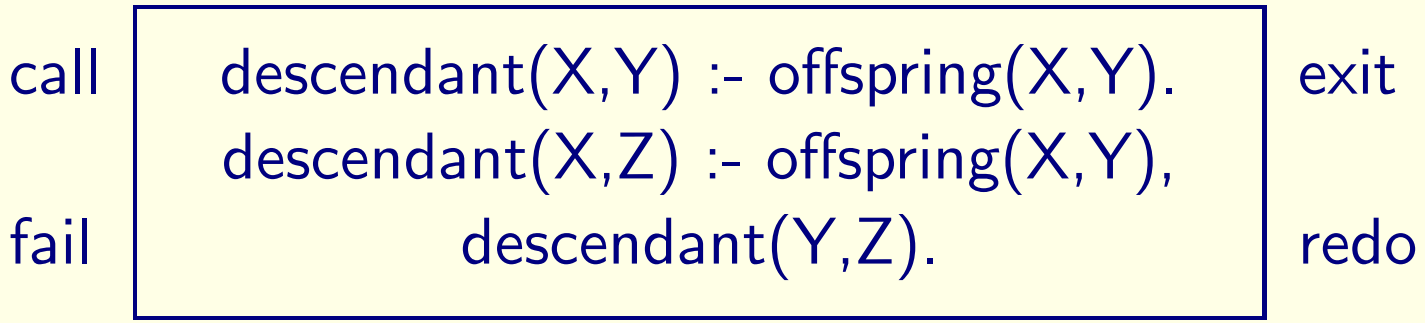
"not affecting the computation" ?

- preserving the success set (what about order etc.?)
- "The transformed program Π' computes wrt the transformed query Q' the same answers and in the same sequence as the original program wrt the original query." (how to specify this?)
- relevant parts of the computation remain the same (...)

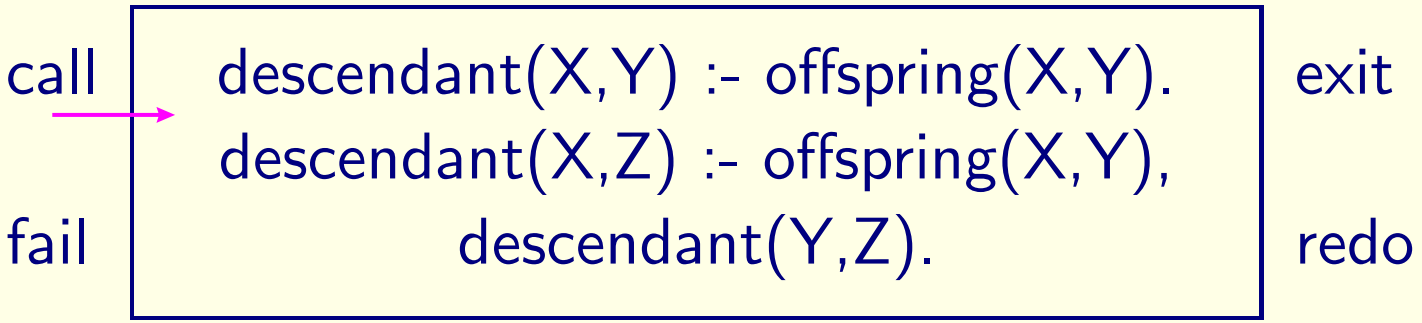
Some problems with describing (even pure) Prolog execution

- complex data flow, alternating in forward and backward direction
- backward steps are much more global than forward steps (the most recent choice point may be any distance away)
- lacking in means of simplification
  e. g. abstraction, compositionality, modularity
- representing "state of computation" by data structures that give us too many special cases for theorem proving
  e. g. stack of stacks of frames (ancestor+environment)

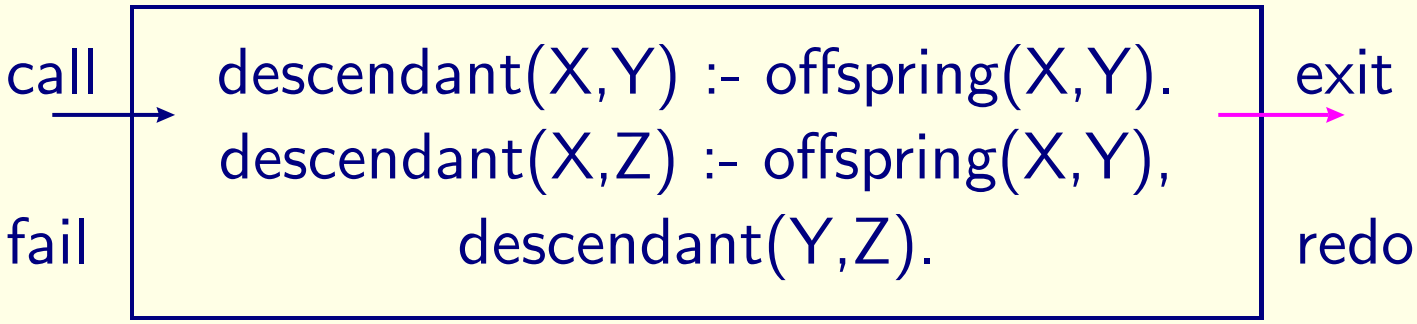
Byrd's "box" metaphor of Prolog execution



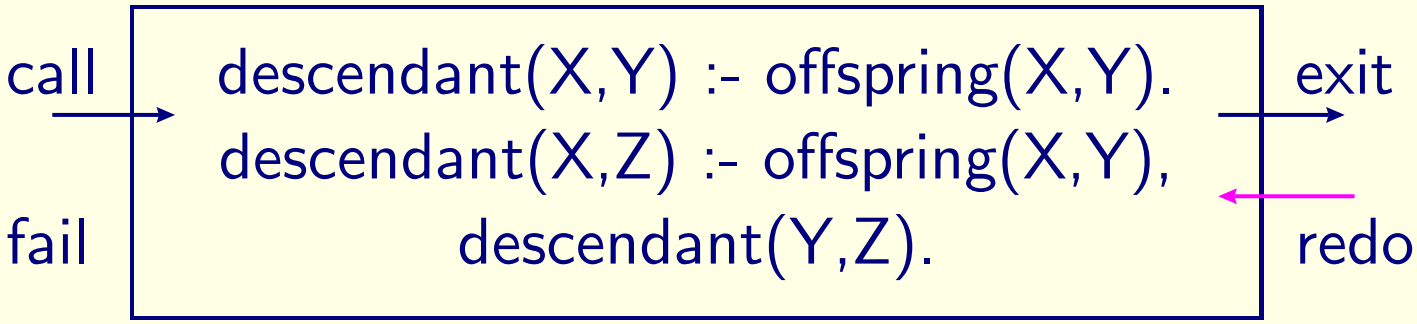
Byrd's "box" metaphor of Prolog execution



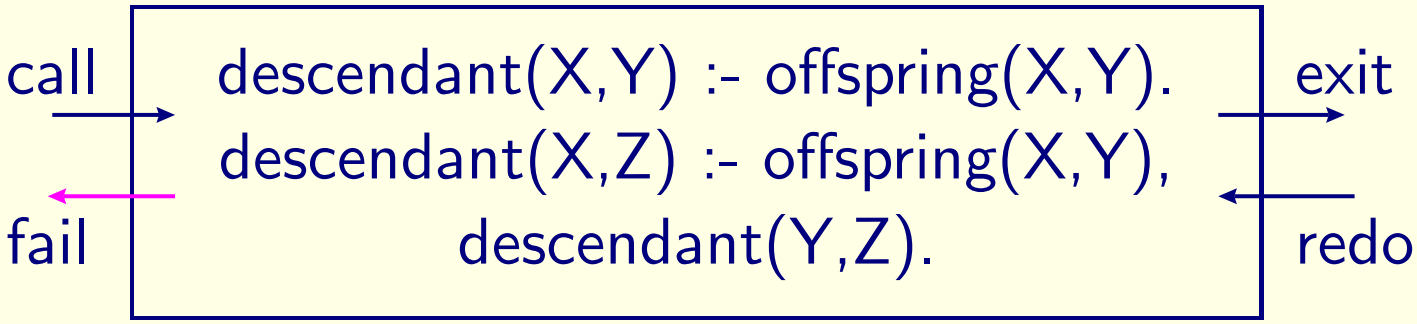
Byrd's "box" metaphor of Prolog execution



Byrd's "box" metaphor of Prolog execution



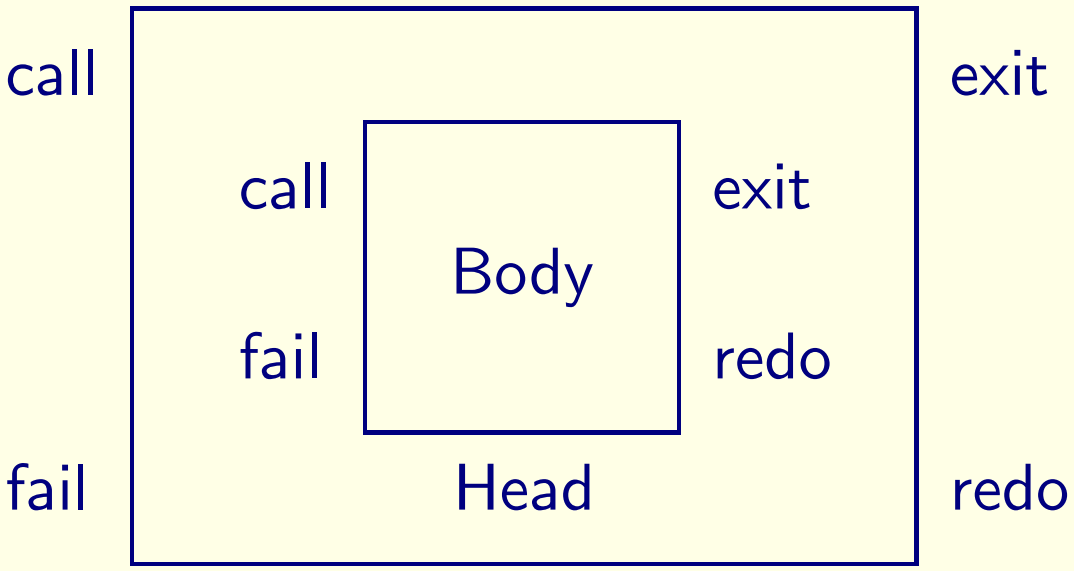
Byrd's "box" metaphor of Prolog execution



PS. What does the box stand for? Does it have to be the "selected atom"?

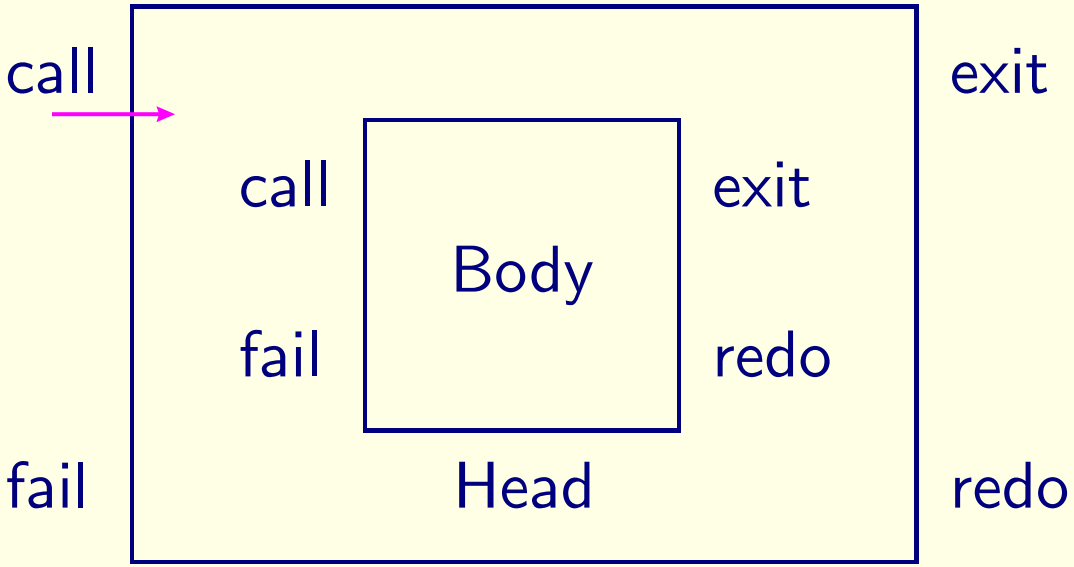
... or can it be a general goal?

- focus on general goals is a main idea behind the S_1:PP model
- canonical form of predicates reduces choice to disjunction



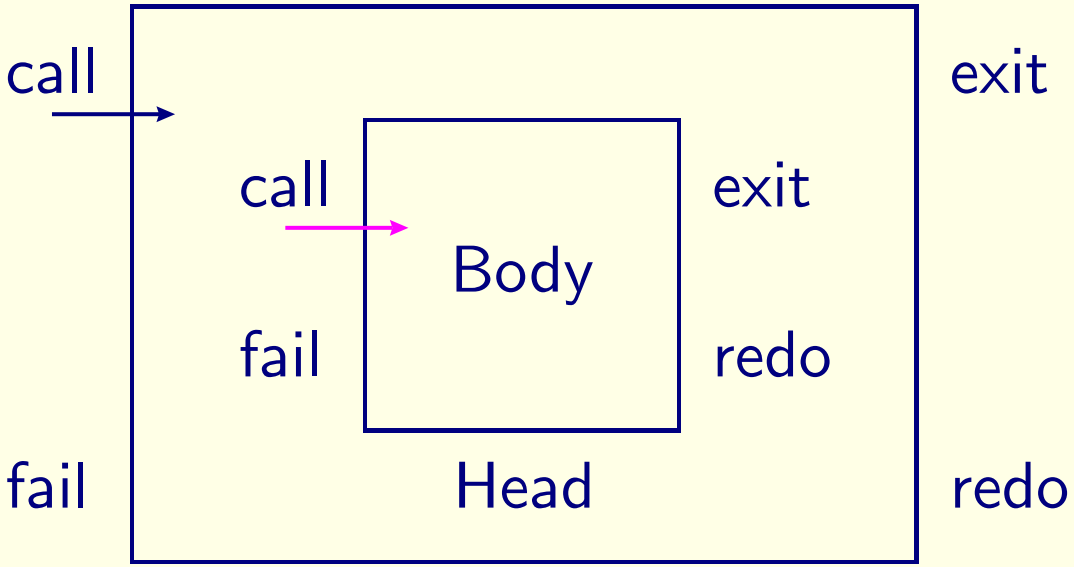
... or can it be a general goal?

- focus on general goals is a main idea behind the S_1:PP model
- canonical form of predicates reduces choice to disjunction



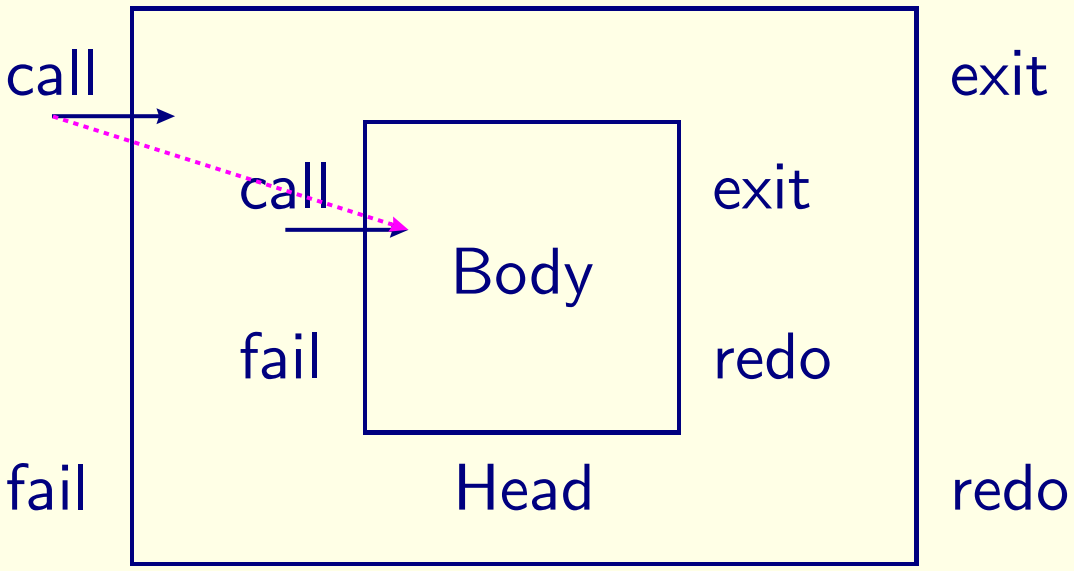
... or can it be a general goal?

- focus on general goals is a main idea behind the S_1:PP model
- canonical form of predicates reduces choice to disjunction



... or can it be a general goal?

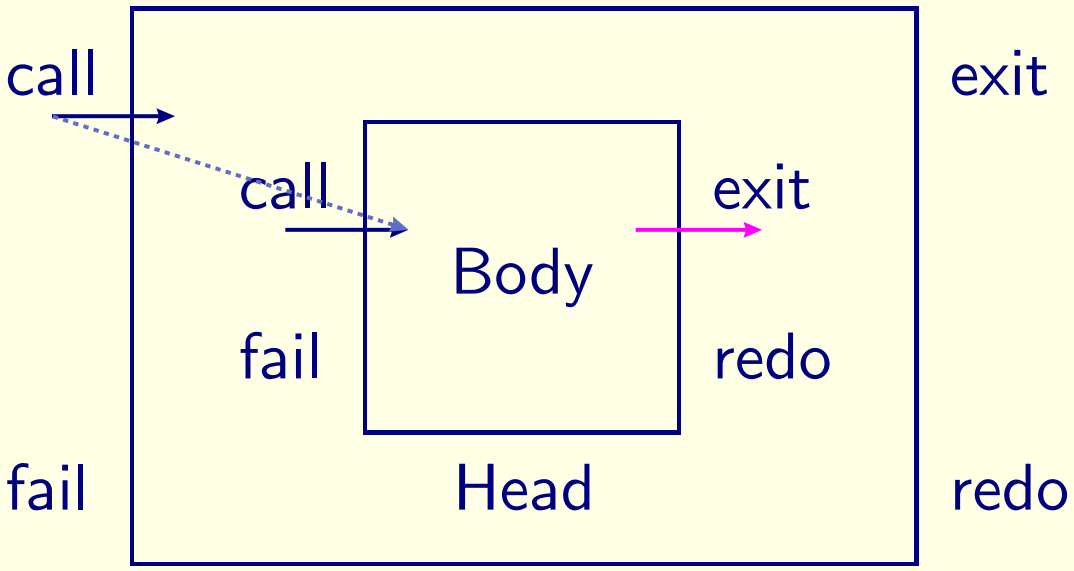
- focus on general goals is a main idea behind the S_1:PP model
- canonical form of predicates reduces choice to disjunction



call Head ⇾ call Body

... or can it be a general goal?

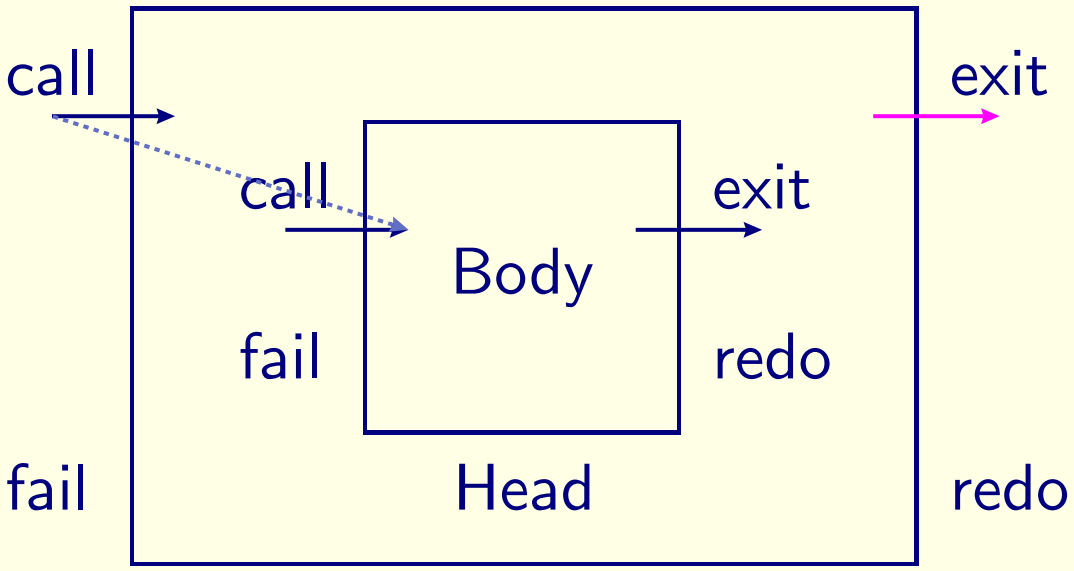
- focus on general goals is a main idea behind the S_1:PP model
- canonical form of predicates reduces choice to disjunction



call Head ⇾ call Body

... or can it be a general goal?

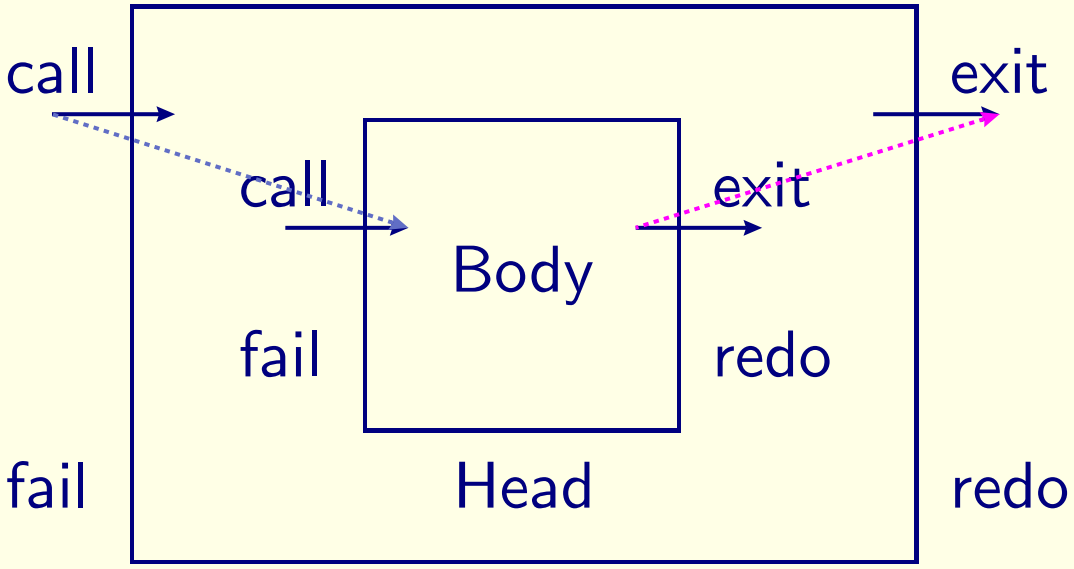
- focus on general goals is a main idea behind the S_1:PP model
- canonical form of predicates reduces choice to disjunction



call Head ⇾ call Body

... or can it be a general goal?

- focus on general goals is a main idea behind the S_1:PP model
- canonical form of predicates reduces choice to disjunction

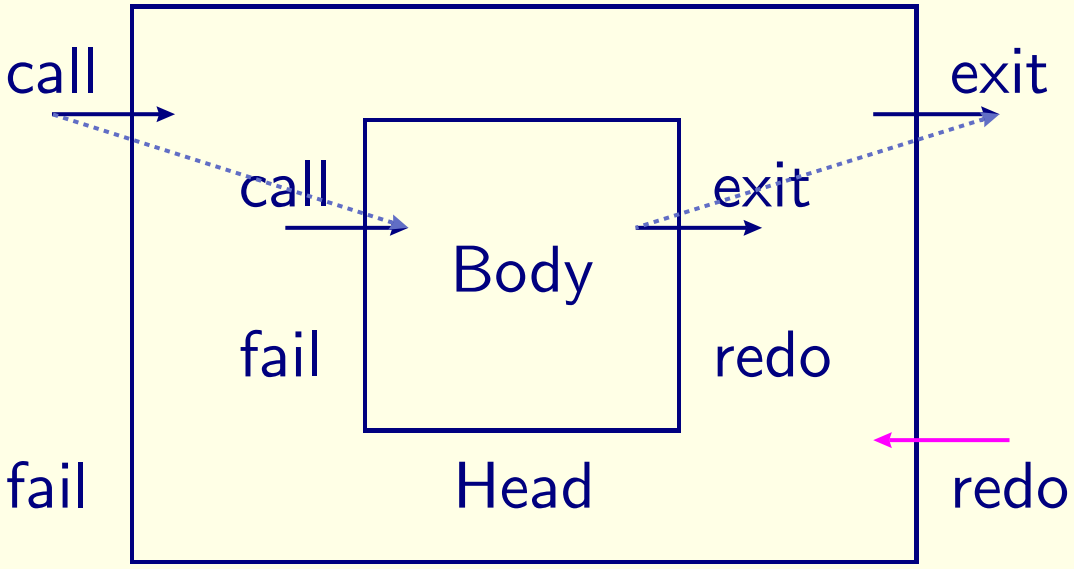


call Head ⇾ call Body

exit Body ⇾ exit Head

... or can it be a general goal?

- focus on general goals is a main idea behind the S_1:PP model
- canonical form of predicates reduces choice to disjunction

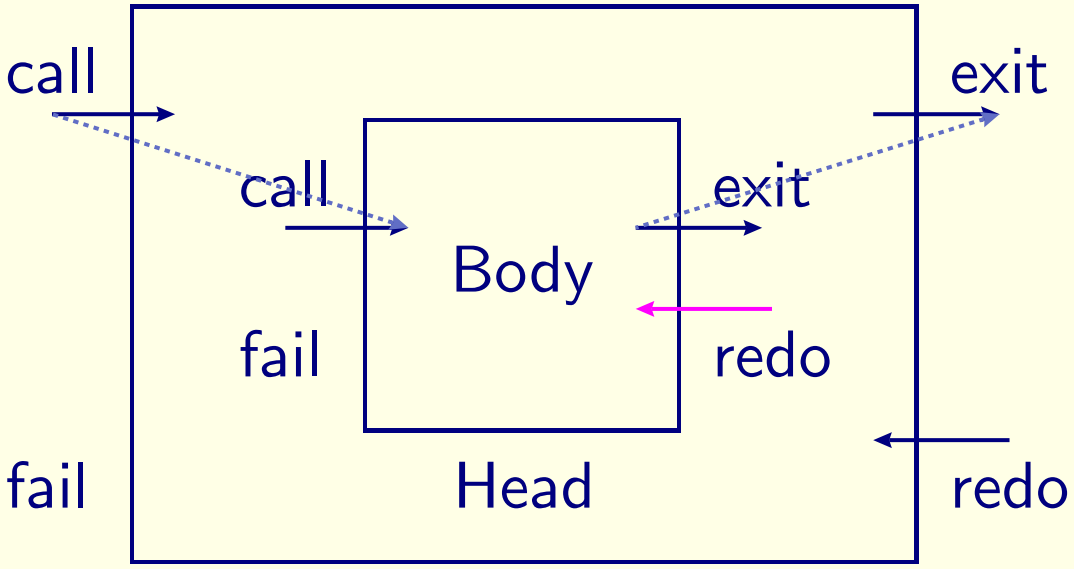


call Head ⇾ call Body

exit Body ⇾ exit Head

... or can it be a general goal?

- focus on general goals is a main idea behind the S_1:PP model
- canonical form of predicates reduces choice to disjunction

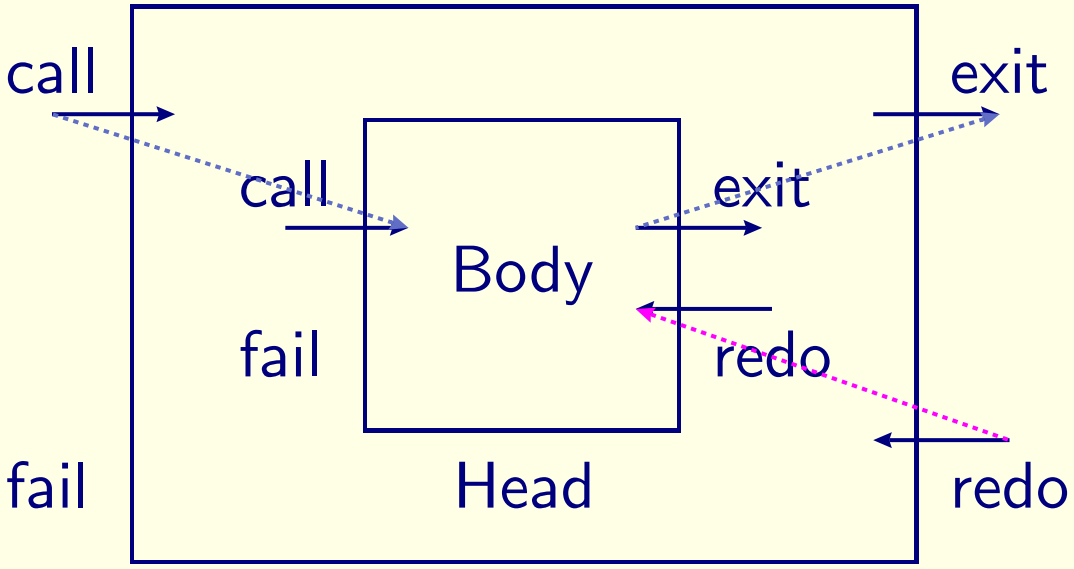


call Head ⇒ call Body

exit Body ⇒ exit Head

... or can it be a general goal?

- focus on general goals is a main idea behind the S_1:PP model
- canonical form of predicates reduces choice to disjunction



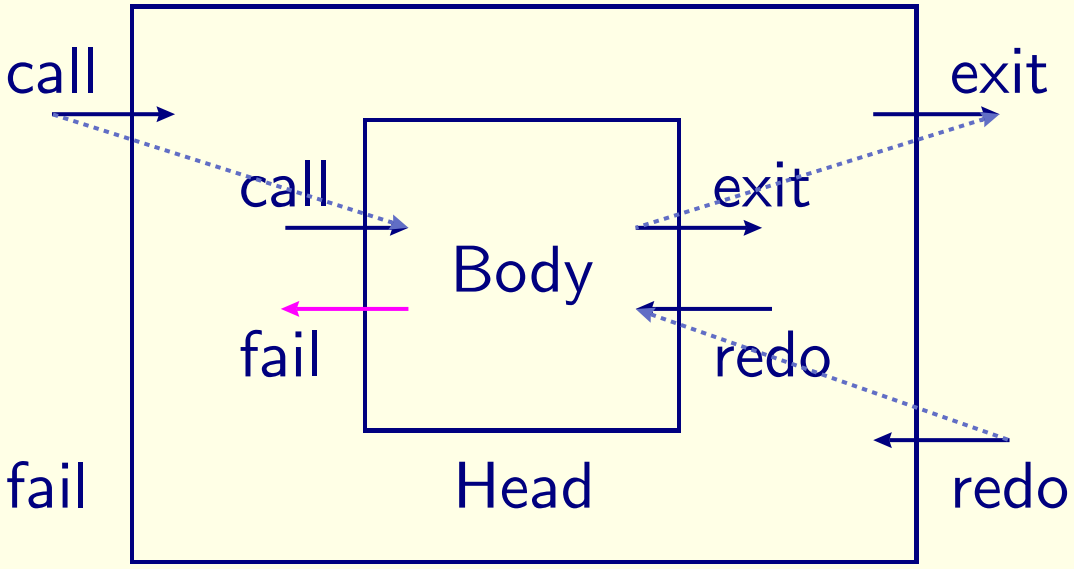
call Head ⇾ call Body

exit Body ⇾ exit Head

redo Head ⇾ redo Body

... or can it be a general goal?

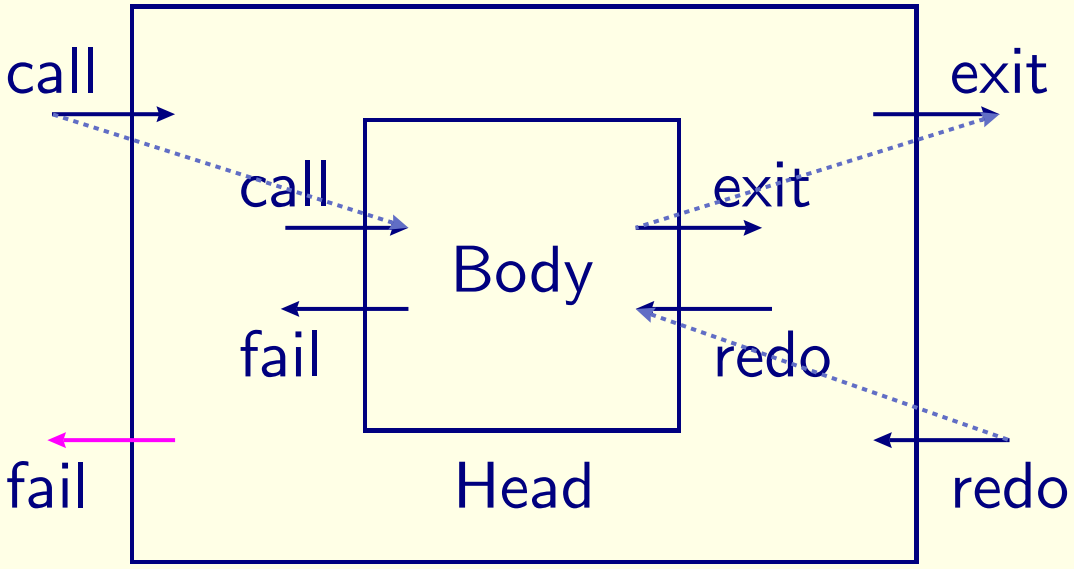
- focus on general goals is a main idea behind the S_1:PP model
- canonical form of predicates reduces choice to disjunction



call Head ⇒ call Body
exit Body ⇒ exit Head
redo Head ⇒ redo Body

... or can it be a general goal?

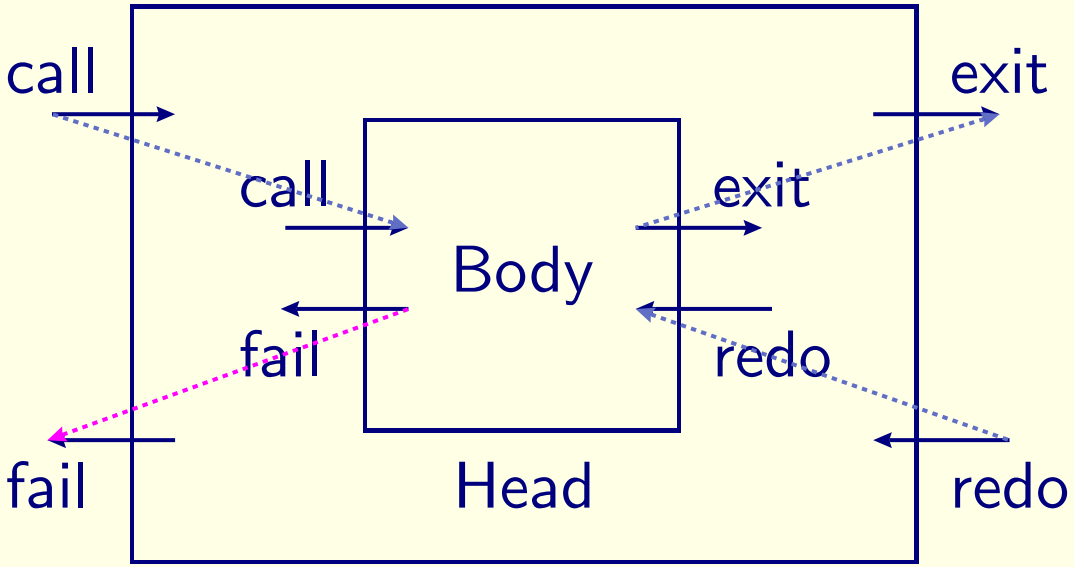
- focus on general goals is a main idea behind the S_1:PP model
- canonical form of predicates reduces choice to disjunction



call Head ⇾ call Body
exit Body ⇾ exit Head
redo Head ⇾ redo Body

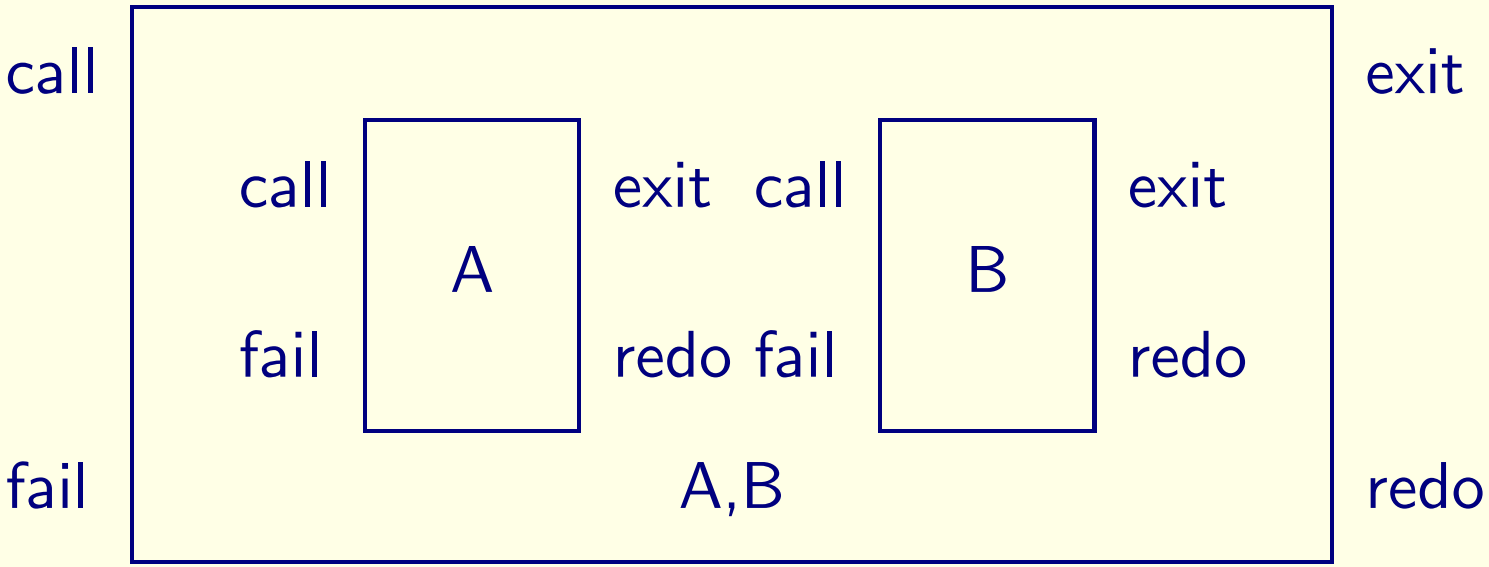
... or can it be a general goal?

- focus on general goals is a main idea behind the S_1:PP model
- canonical form of predicates reduces choice to disjunction

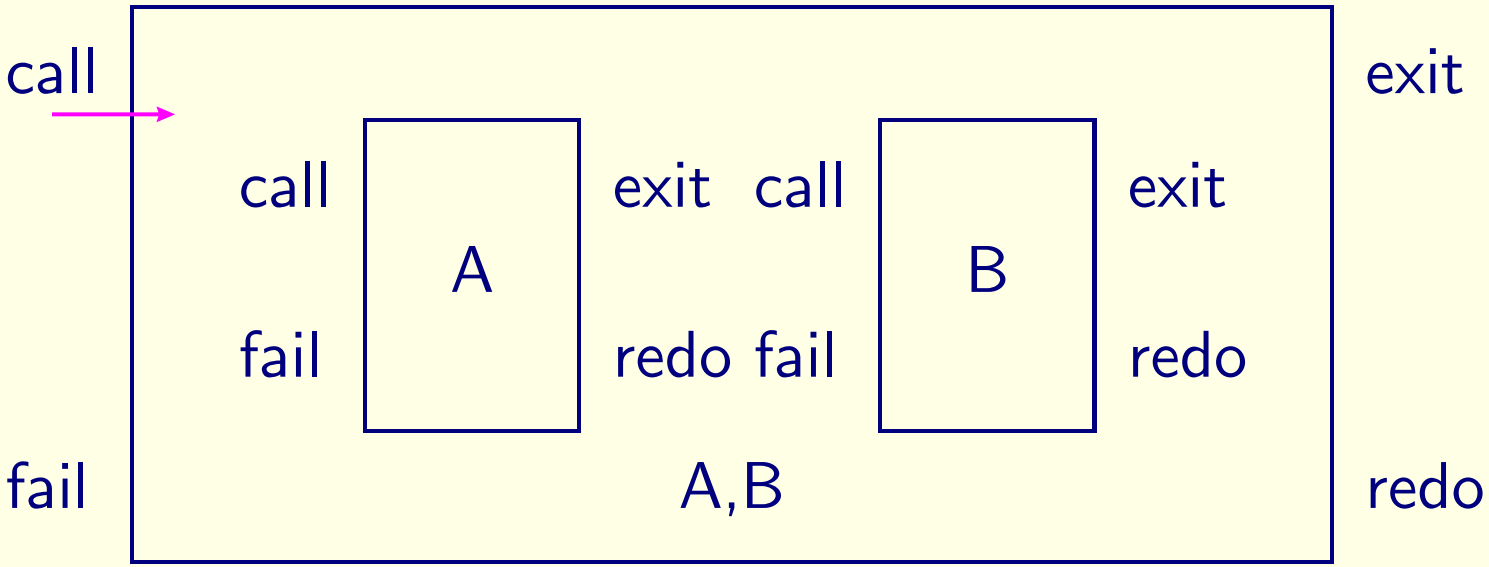


call Head ⇾ call Body
exit Body ⇾ exit Head
redo Head ⇾ redo Body
fail Body ⇾ fail Head

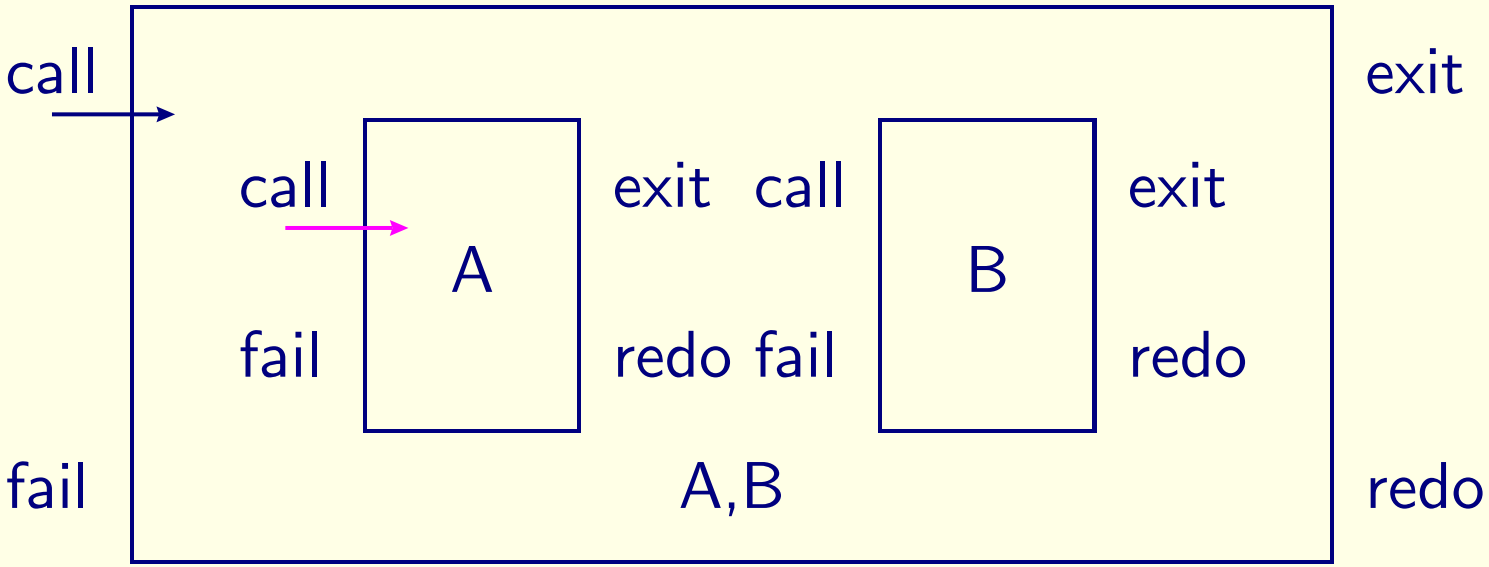
Main idea (cont'd), Conjunction



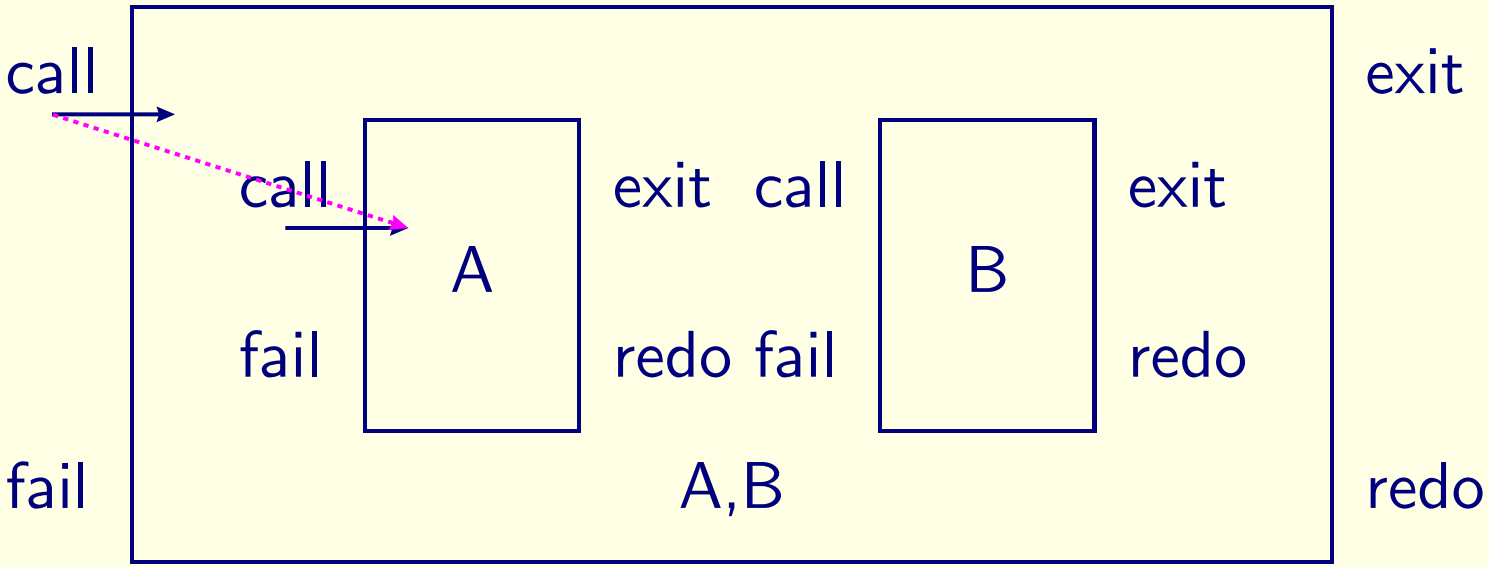
Main idea (cont'd), Conjunction



Main idea (cont'd), Conjunction

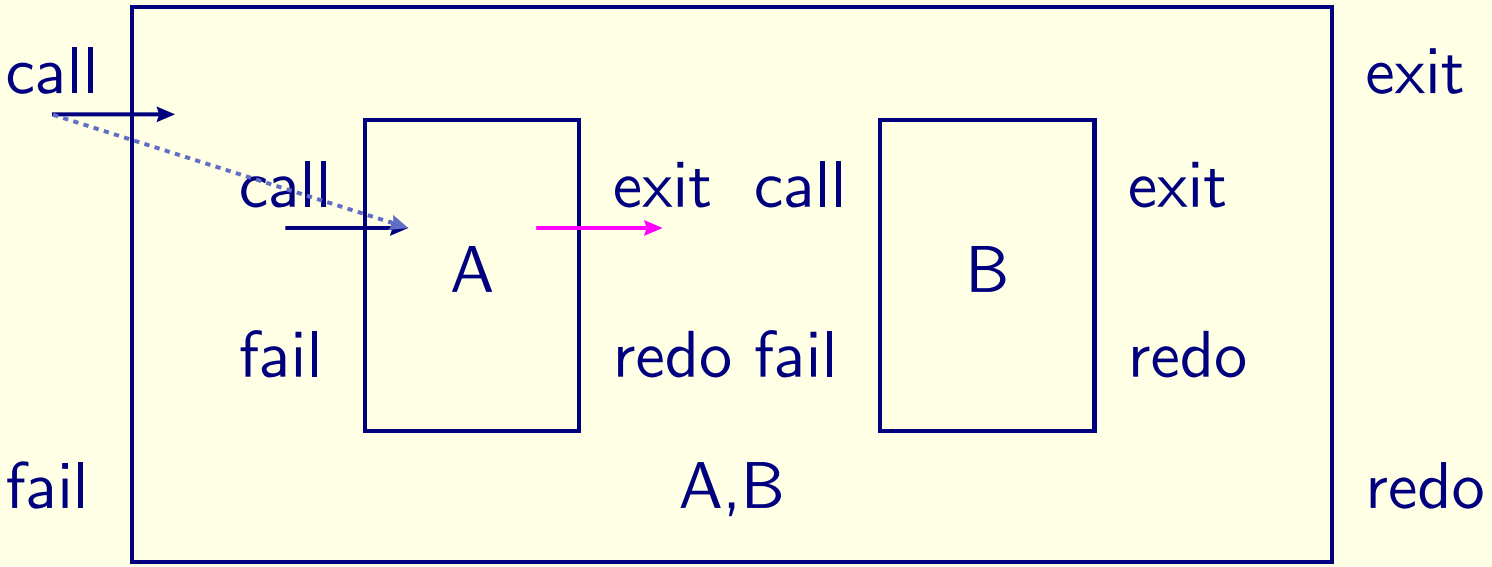


Main idea (cont'd), Conjunction



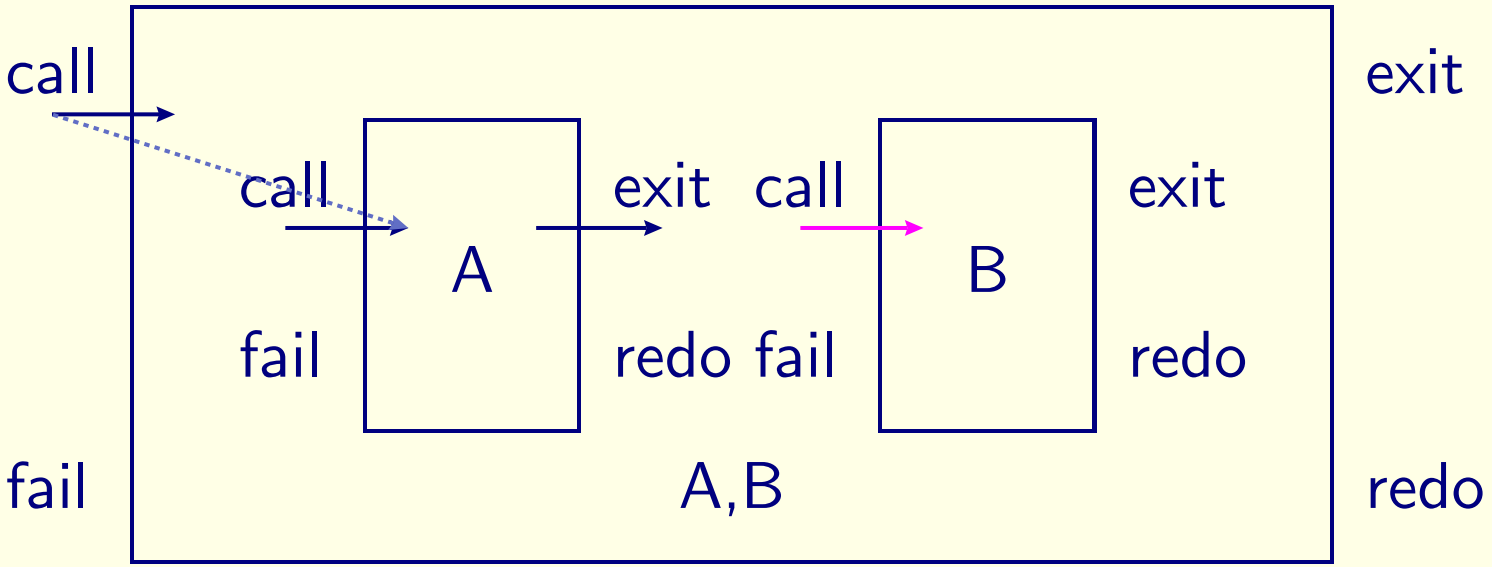
call A,B $\rightarrowtriangle$ call A

Main idea (cont'd), Conjunction



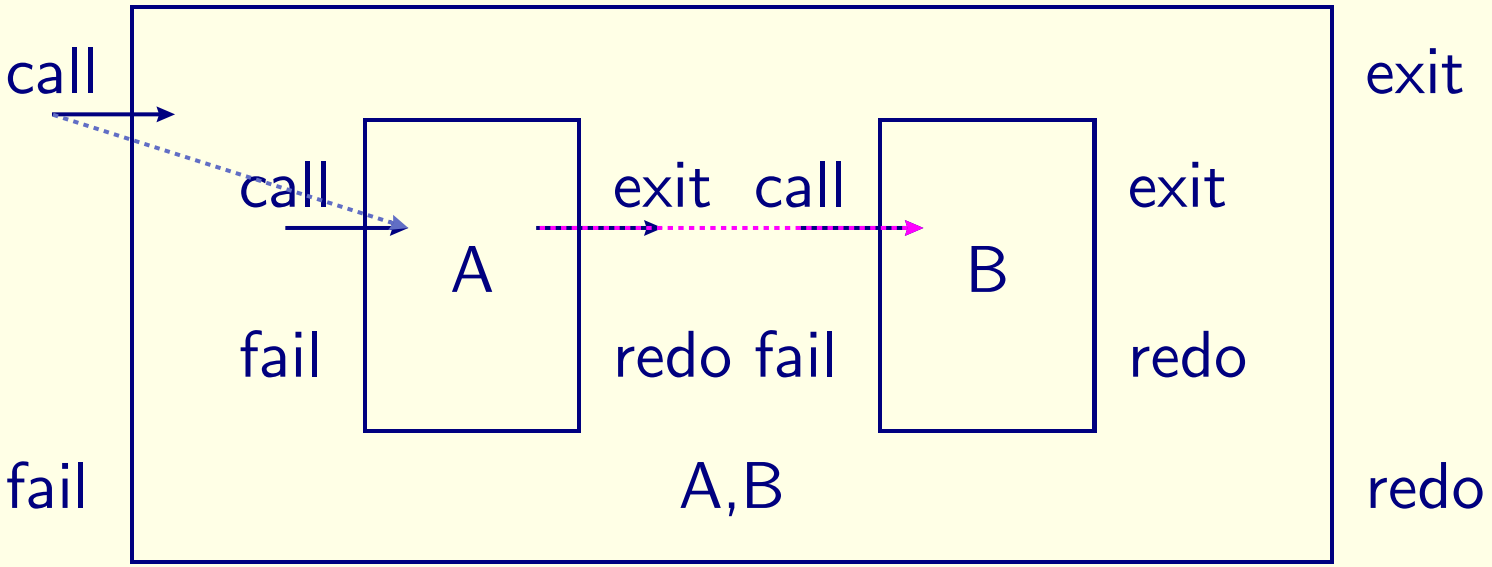
call A,B ⇀ call A

Main idea (cont'd), Conjunction



call A,B ⇀ call A

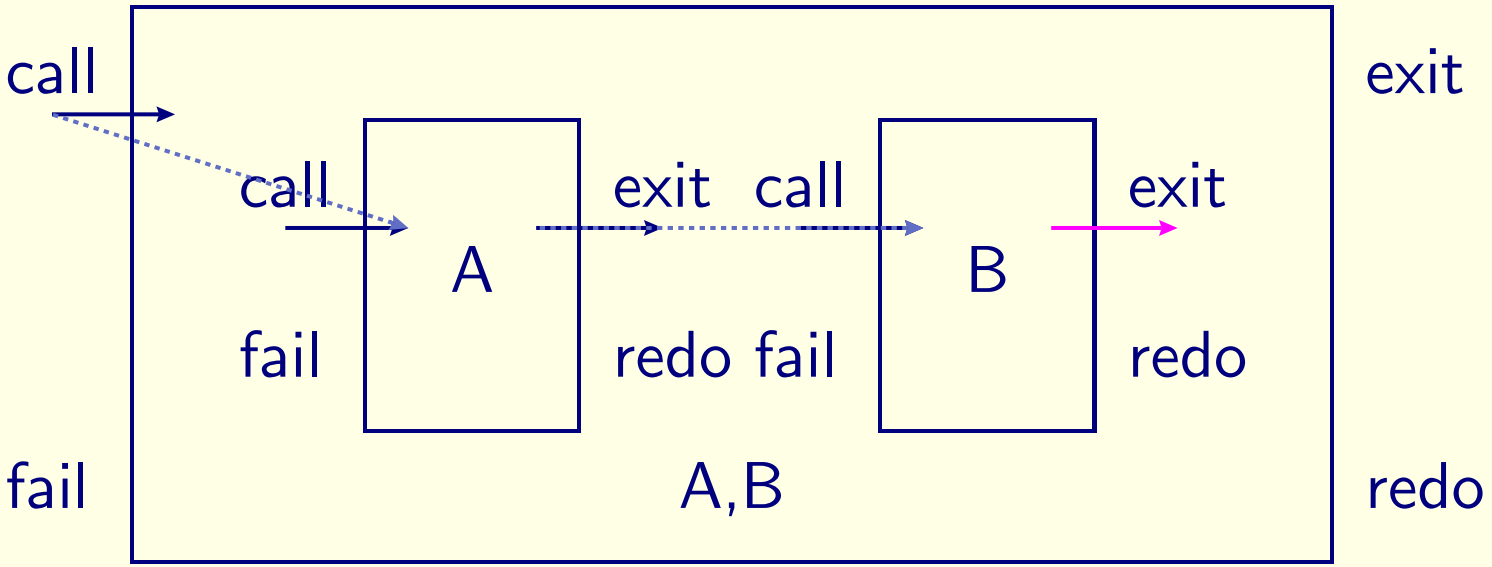
Main idea (cont'd), Conjunction



call A,B → call A

exit A → call B

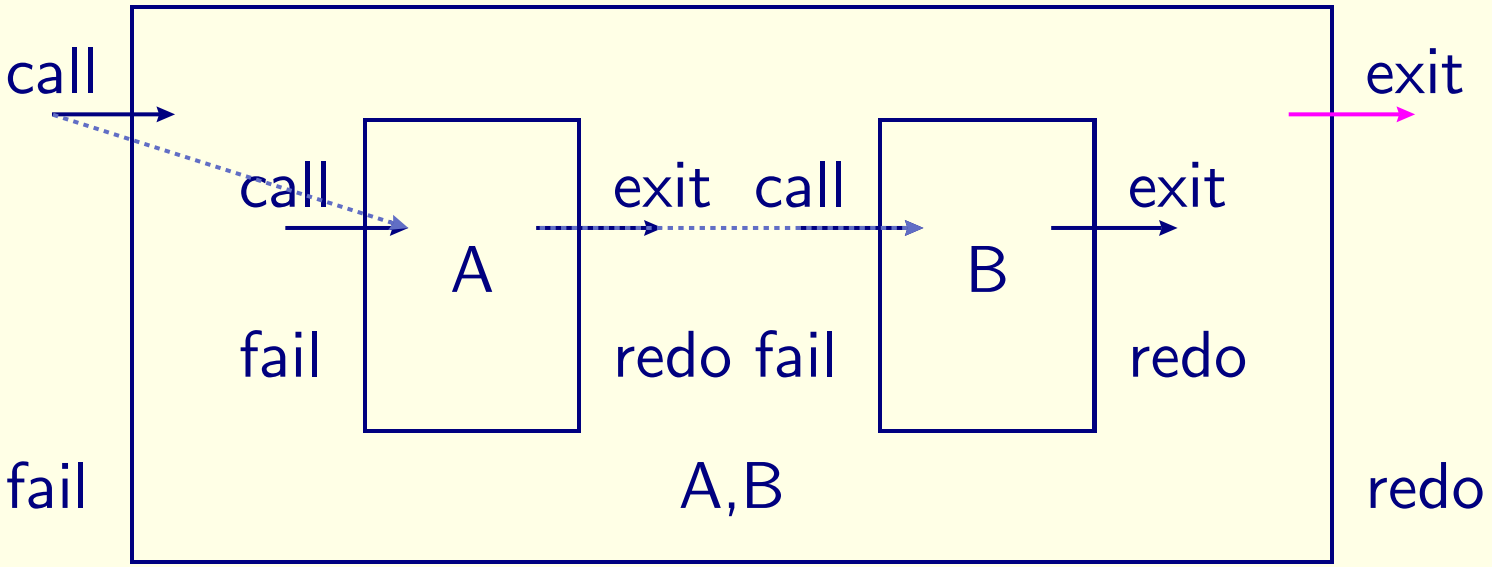
Main idea (cont'd), Conjunction



call A,B → call A

exit A → call B

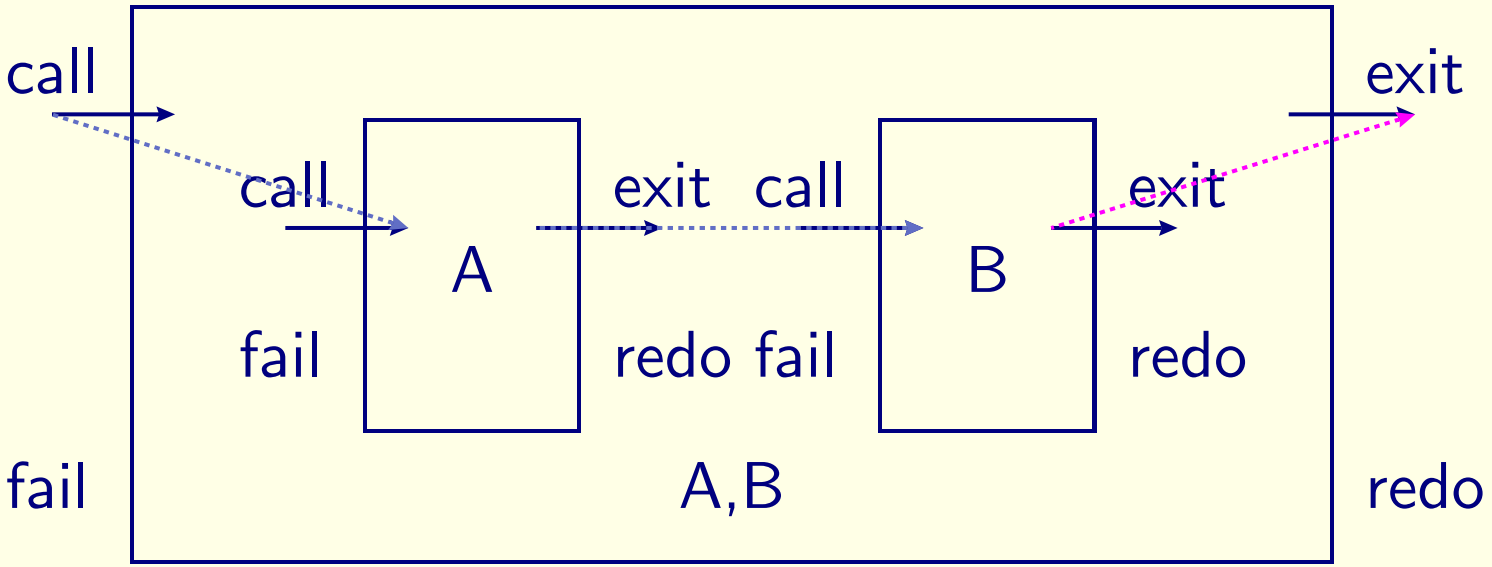
Main idea (cont'd), Conjunction



call A,B → call A

exit A → call B

Main idea (cont'd), Conjunction

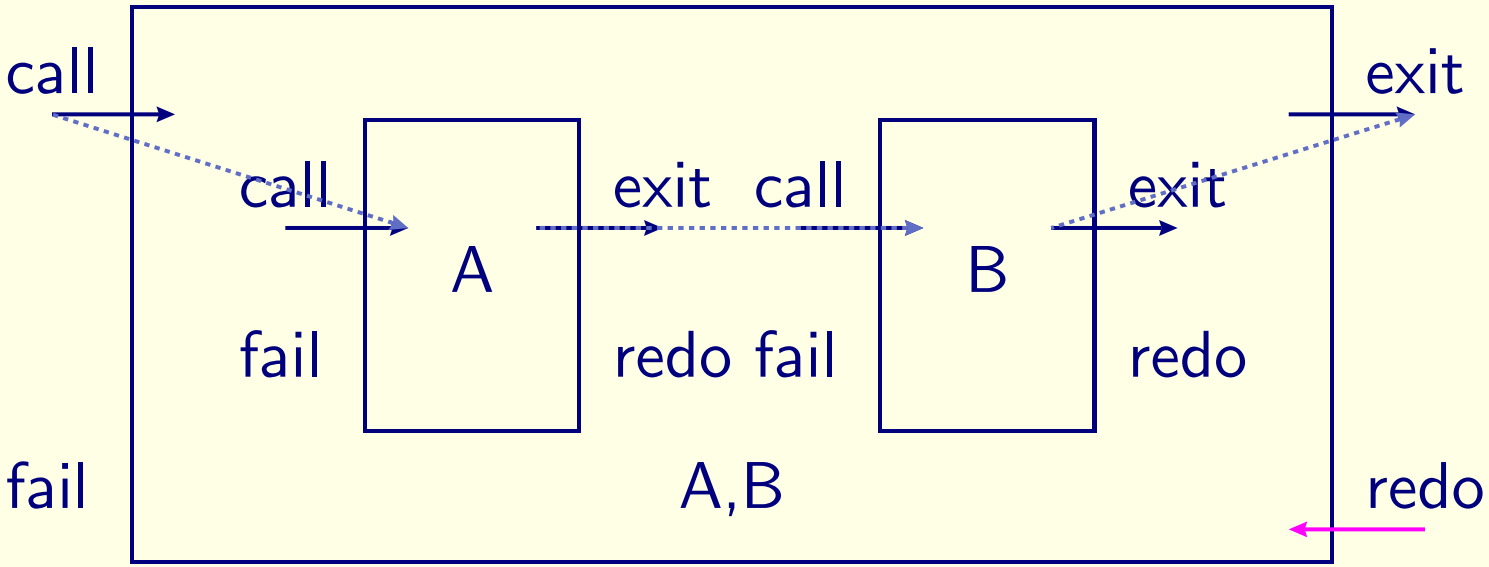


call A,B → call A

exit A → call B

exit B → exit A,B

Main idea (cont'd), Conjunction

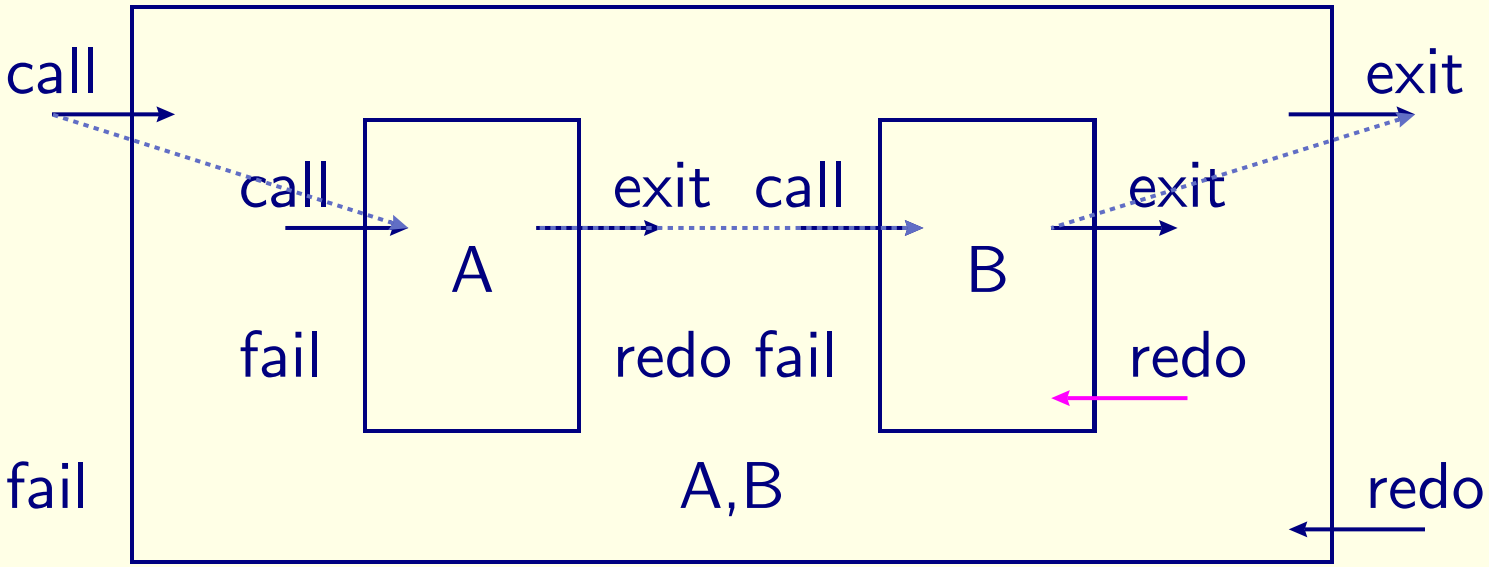


call A,B → call A

exit A → call B

exit B → exit A,B

Main idea (cont'd), Conjunction

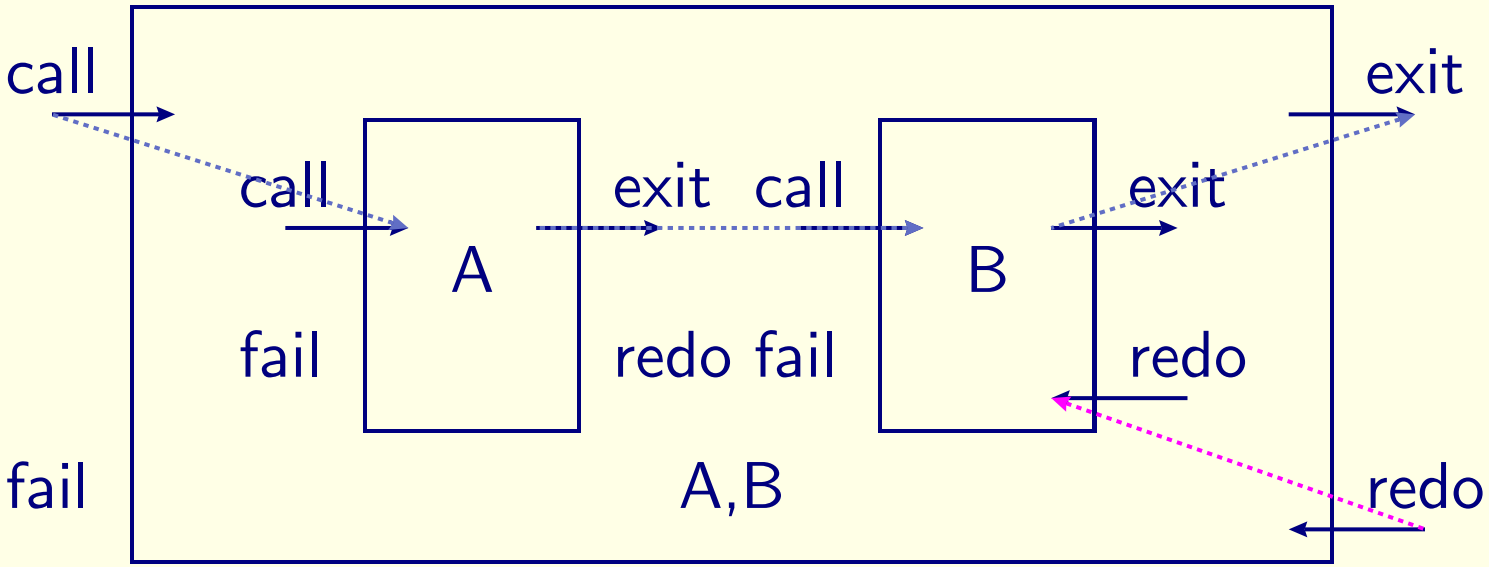


call A,B → call A

exit A → call B

exit B → exit A,B

Main idea (cont'd), Conjunction



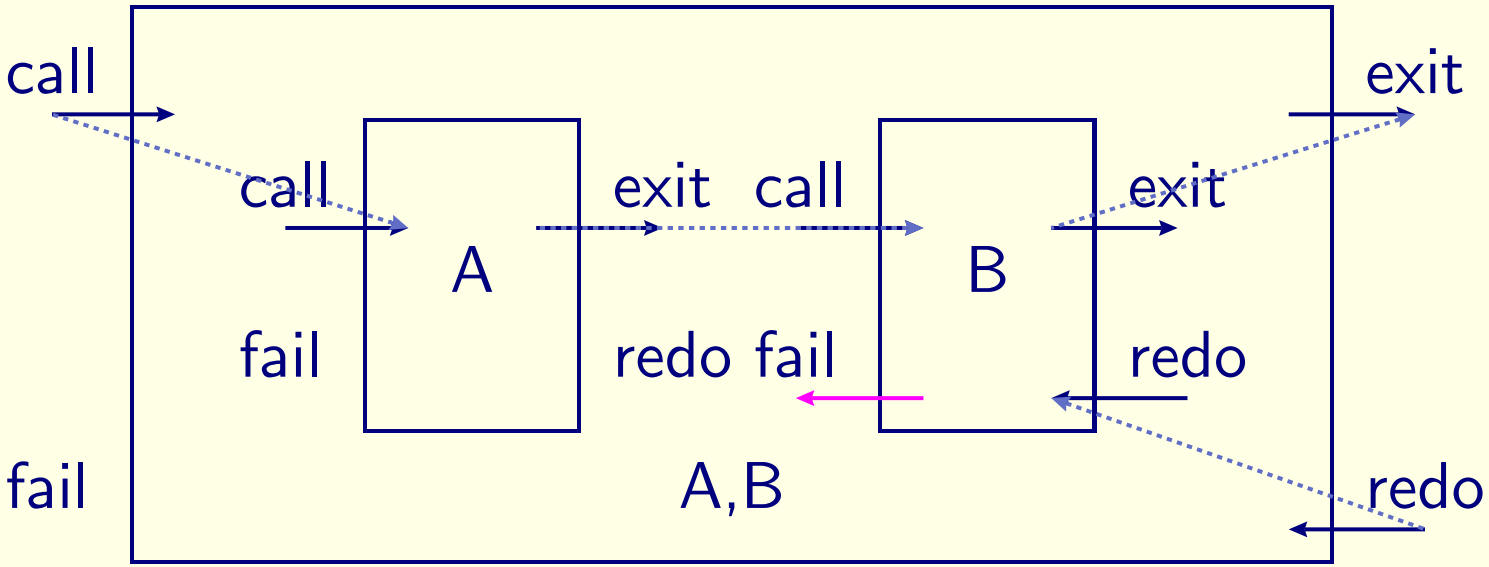
call A,B → call A

exit A → call B

exit B → exit A,B

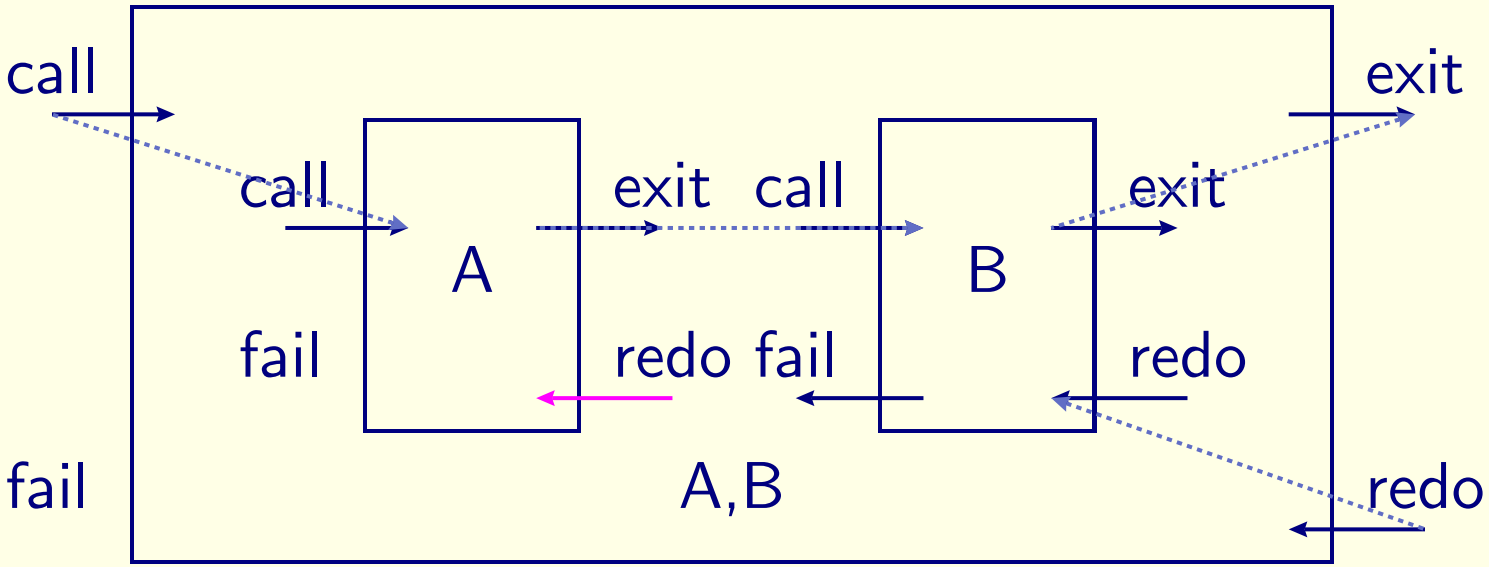
redo A,B → redo B

Main idea (cont'd), Conjunction



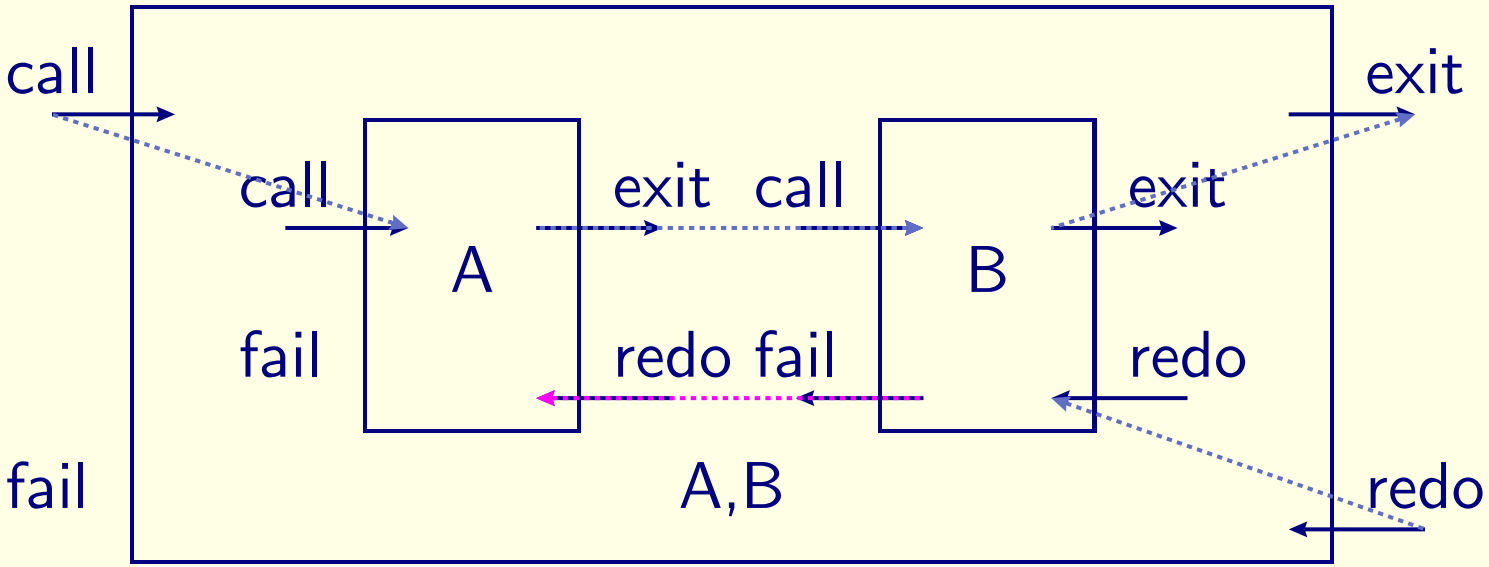
call A,B	→	call A
exit A	→	call B
exit B	→	exit A,B
redo A,B	→	redo B

Main idea (cont'd), Conjunction



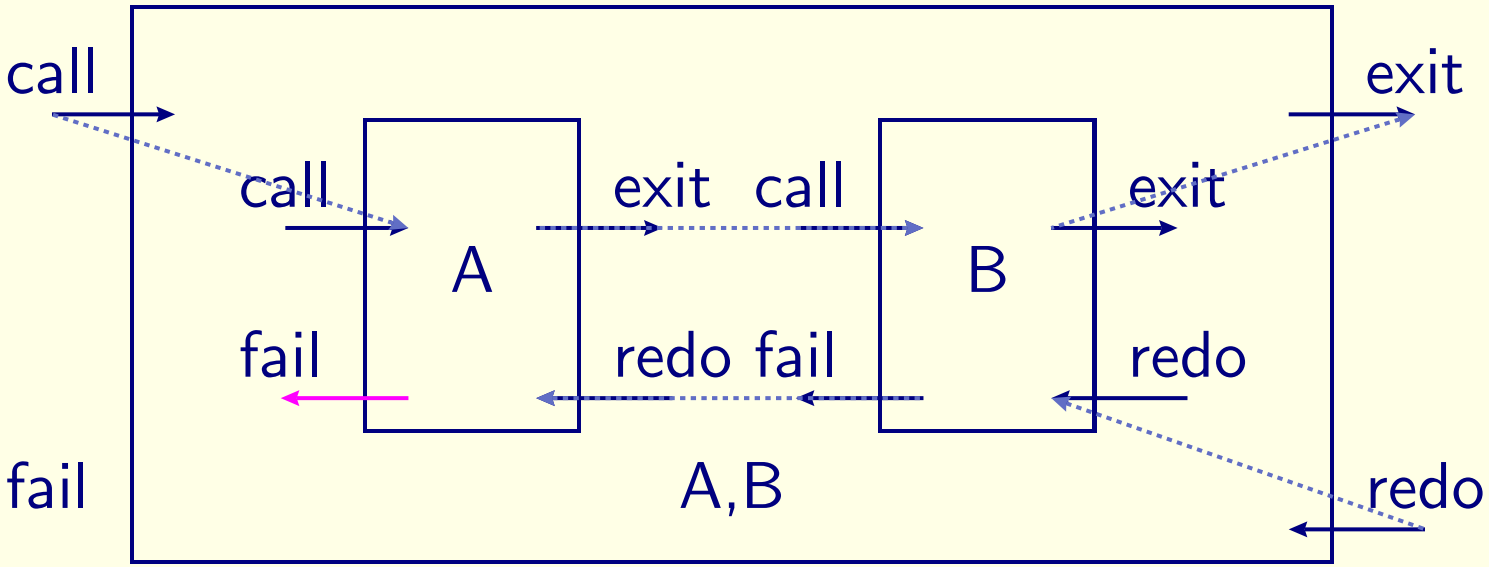
call A,B	→	call A
exit A	→	call B
exit B	→	exit A,B
redo A,B	→	redo B

Main idea (cont'd), Conjunction



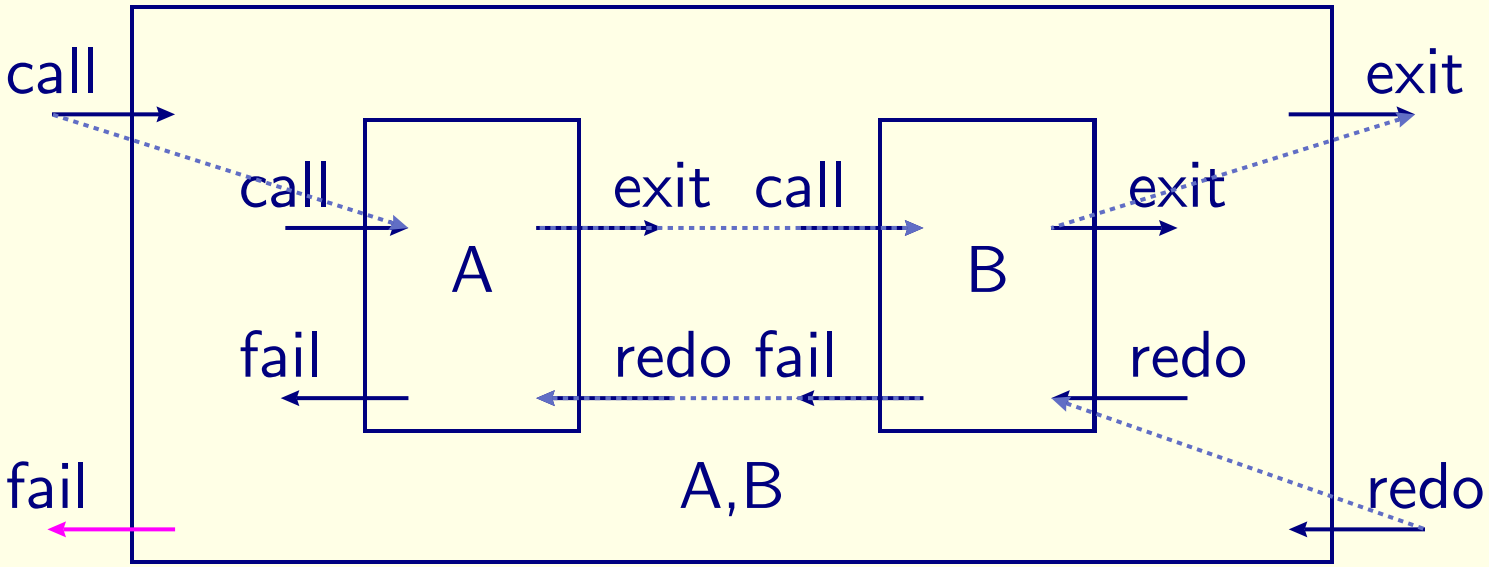
call A,B	→	call A
exit A	→	call B
exit B	→	exit A,B
redo A,B	→	redo B
fail B	→	redo A

Main idea (cont'd), Conjunction



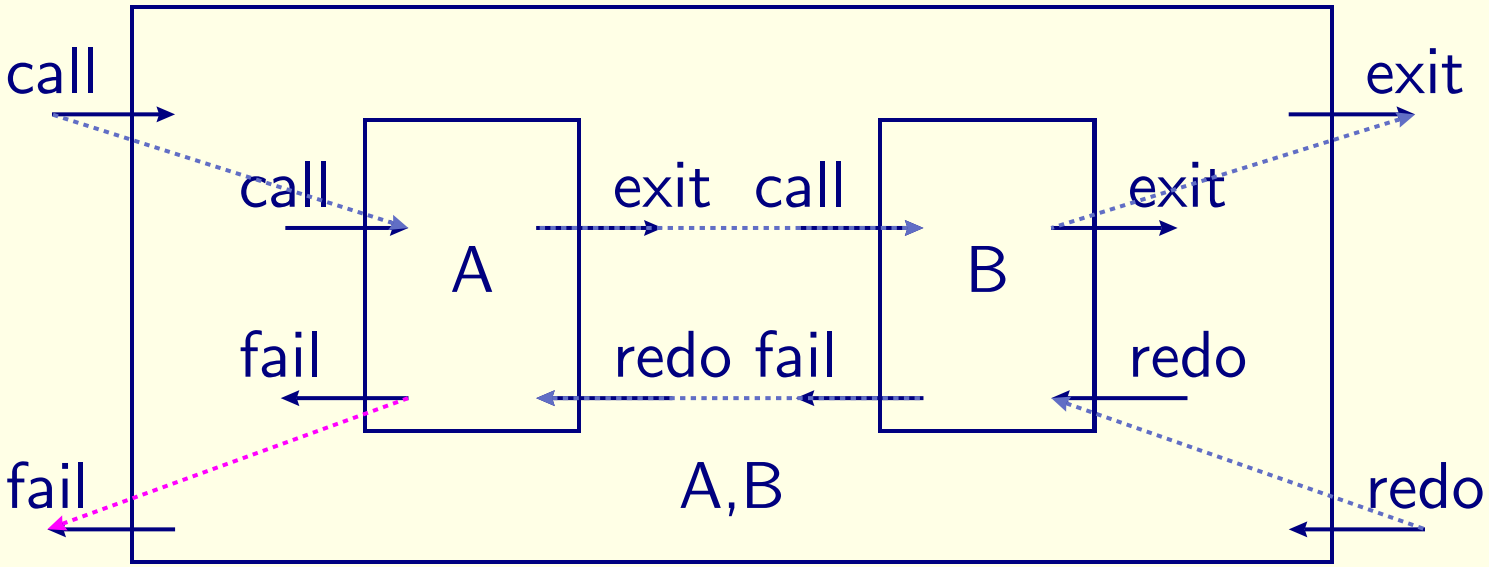
call A,B	→	call A
exit A	→	call B
exit B	→	exit A,B
redo A,B	→	redo B
fail B	→	redo A

Main idea (cont'd), Conjunction



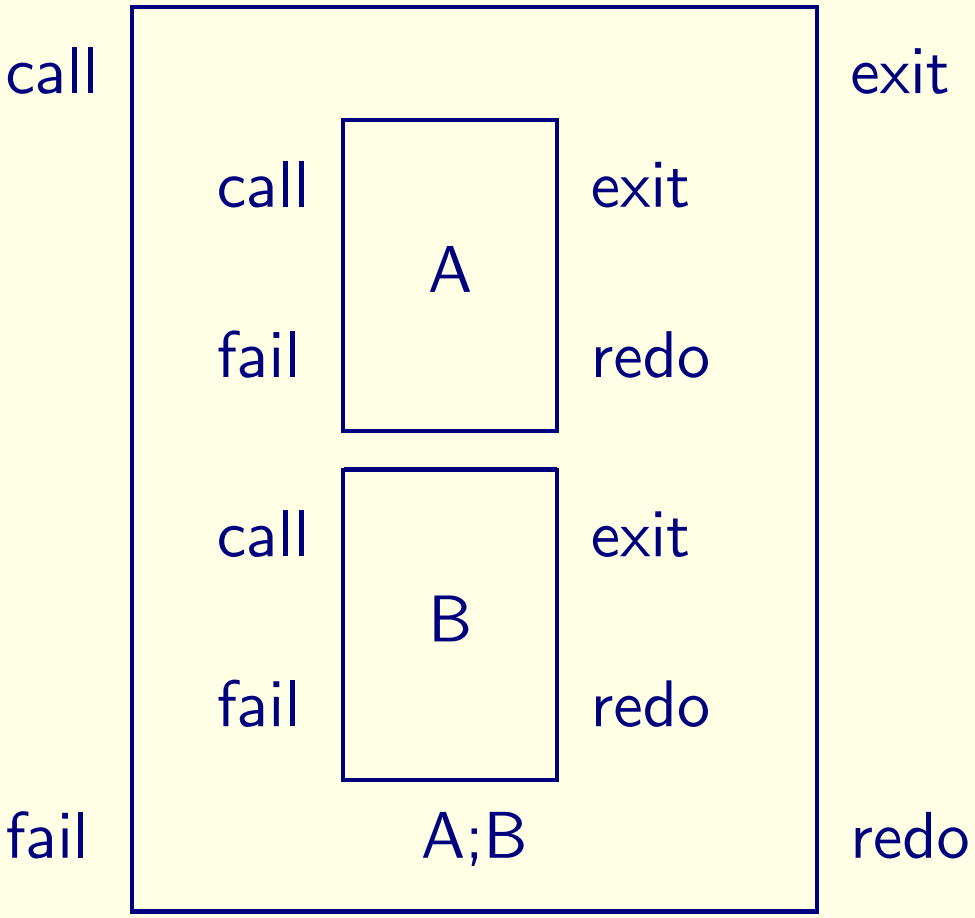
call A,B	→	call A
exit A	→	call B
exit B	→	exit A,B
redo A,B	→	redo B
fail B	→	redo A

Main idea (cont'd), Conjunction

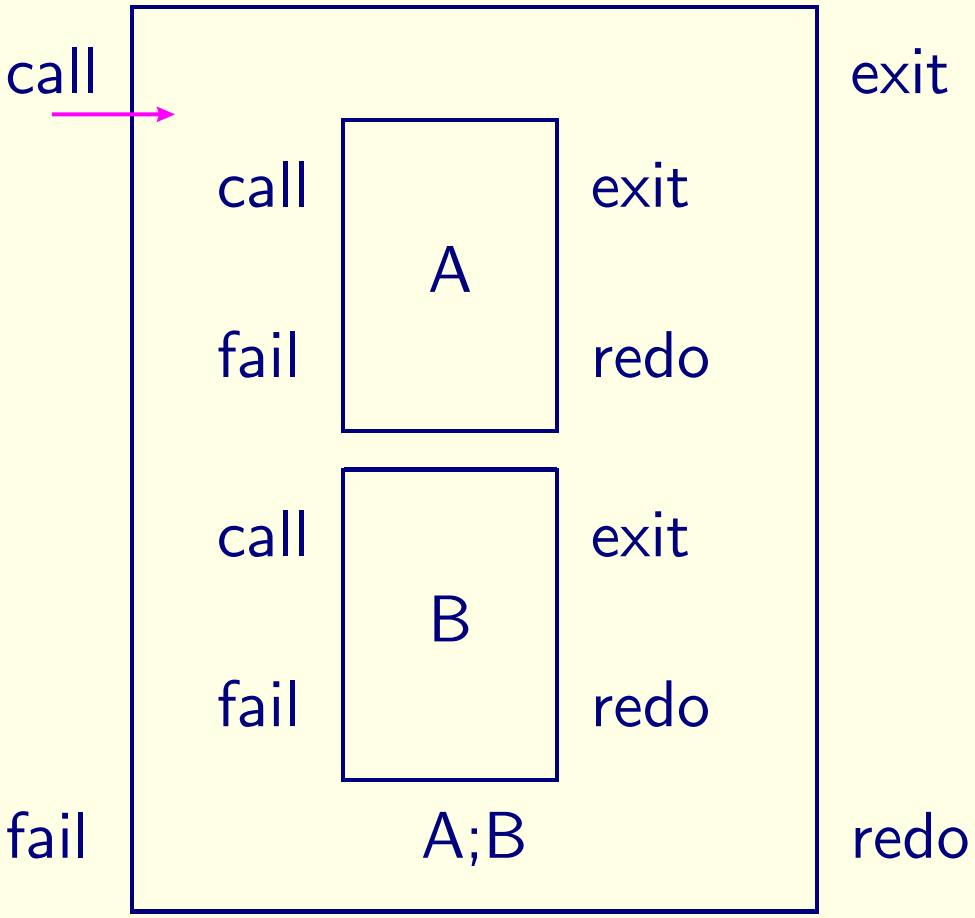


call A,B	→	call A
exit A	→	call B
exit B	→	exit A,B
redo A,B	→	redo B
fail B	→	redo A
fail A	→	fail A,B

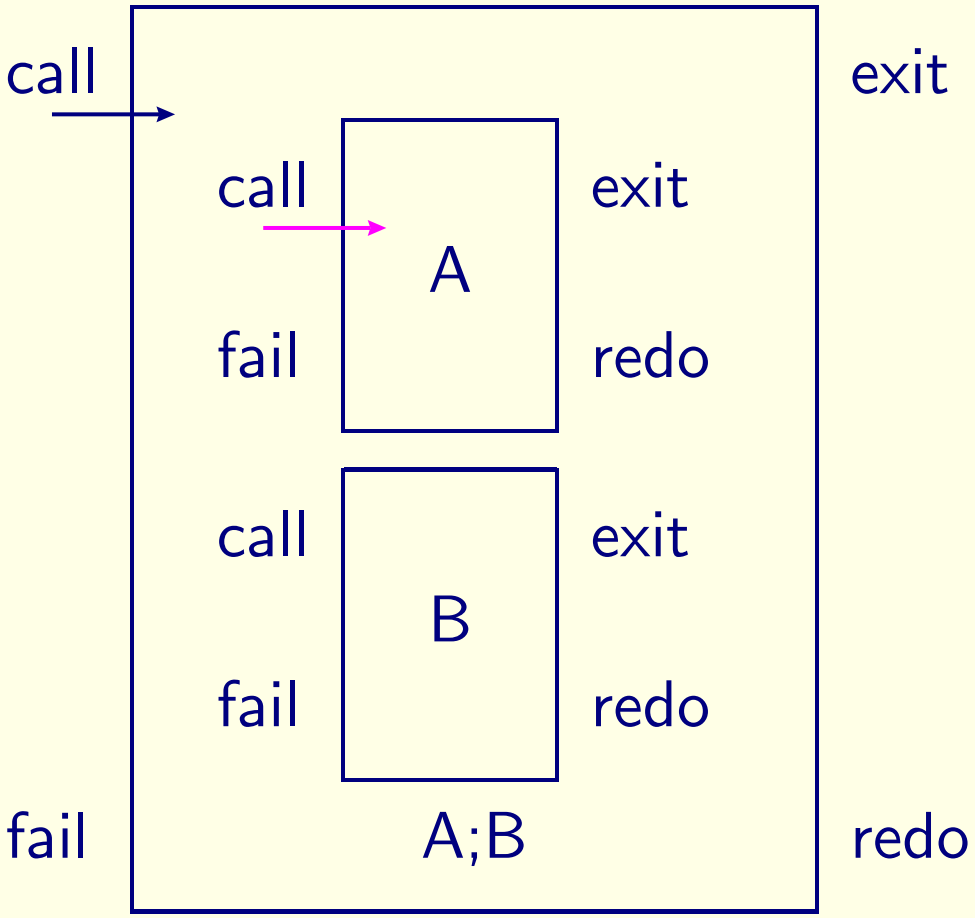
Main idea (cont'd), Disjunction



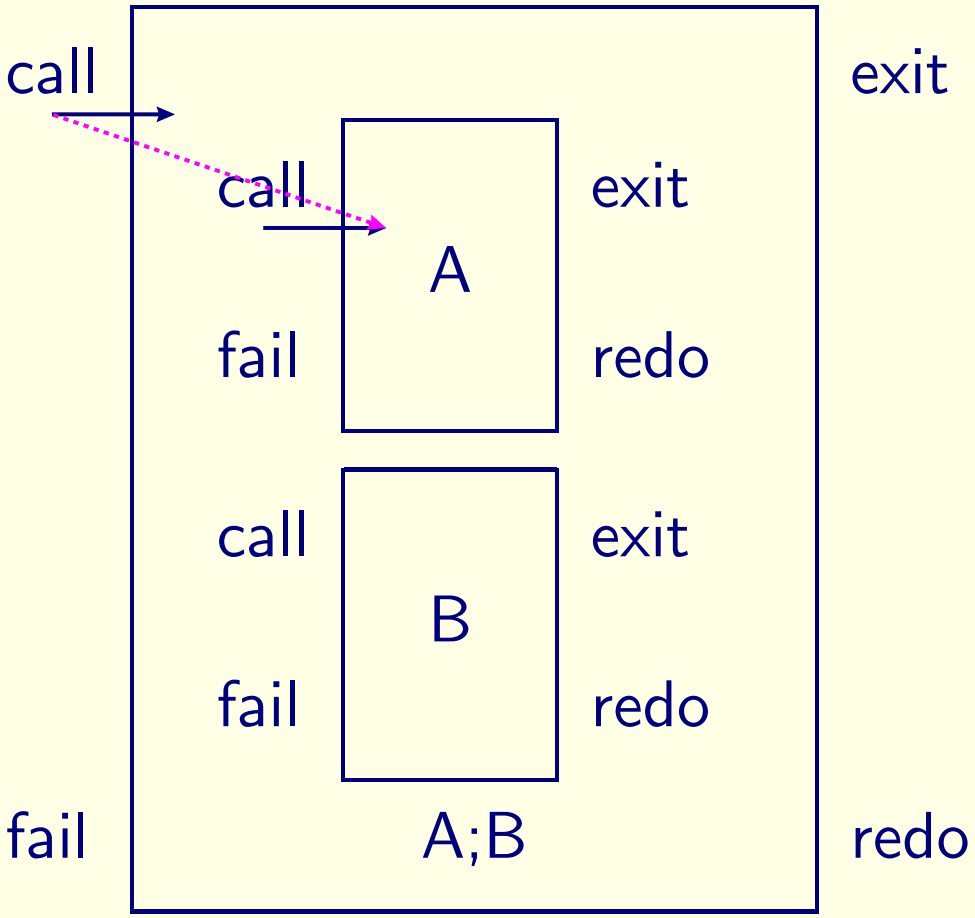
Main idea (cont'd), Disjunction



Main idea (cont'd), Disjunction

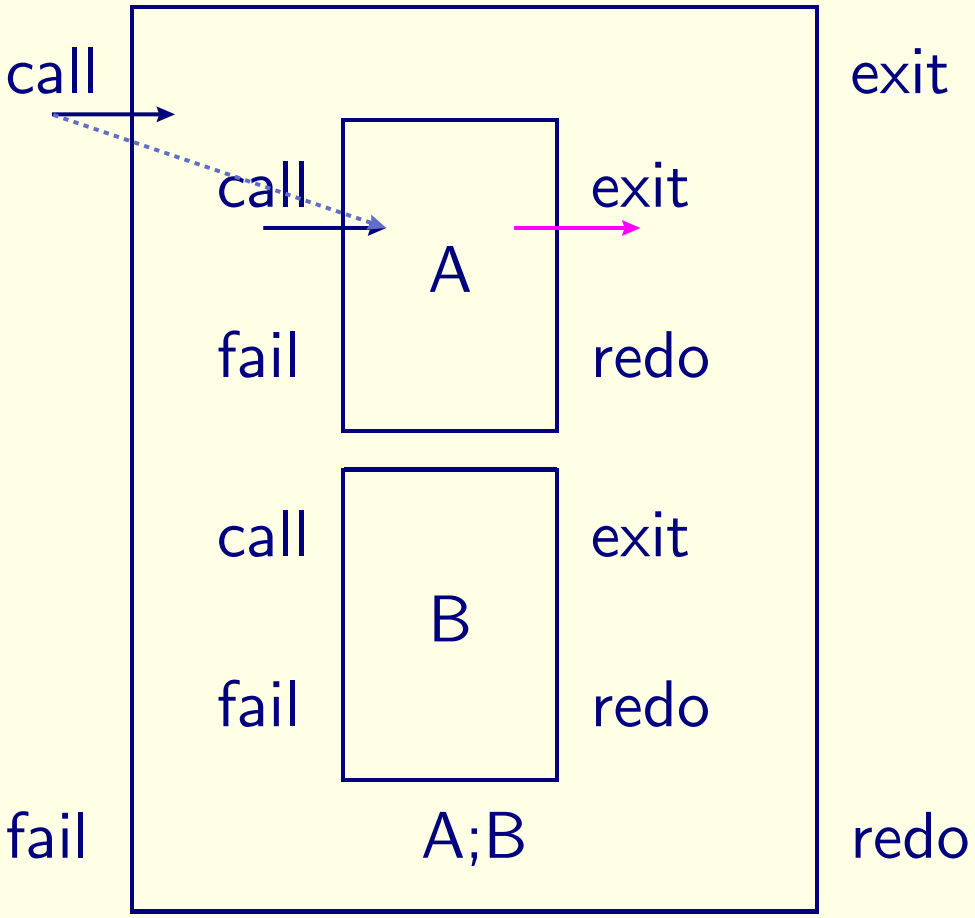


Main idea (cont'd), Disjunction



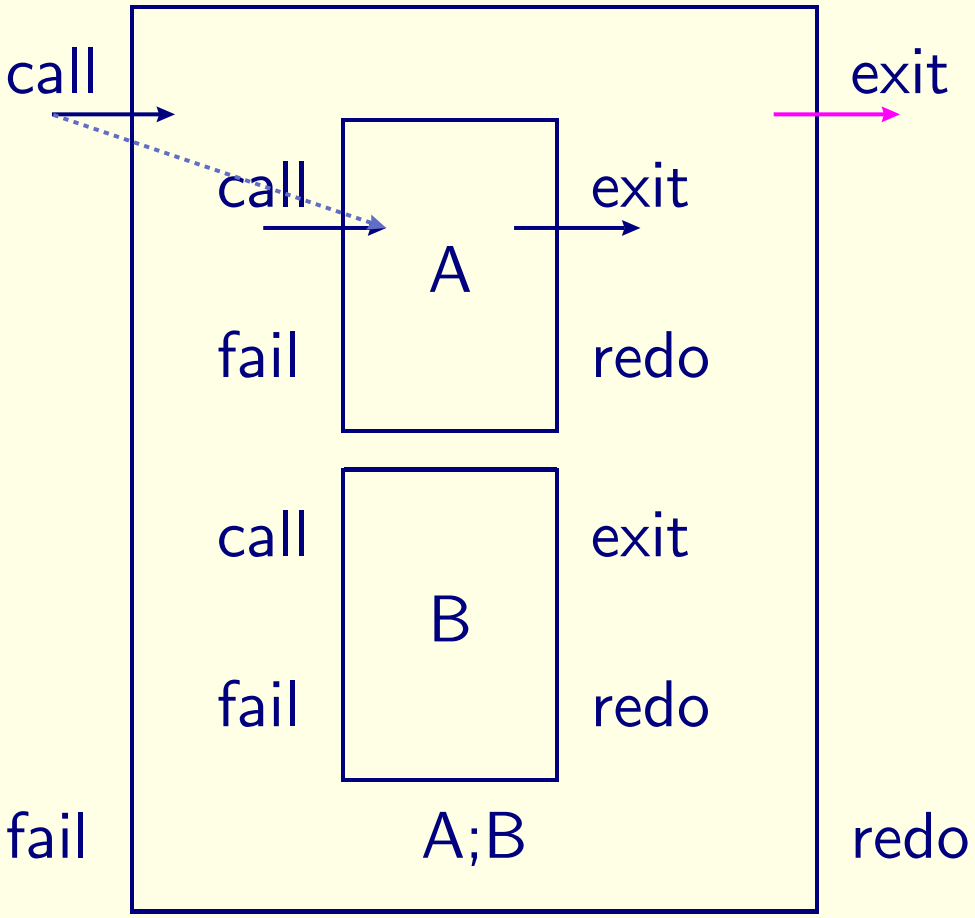
call A;B → call A

Main idea (cont'd), Disjunction



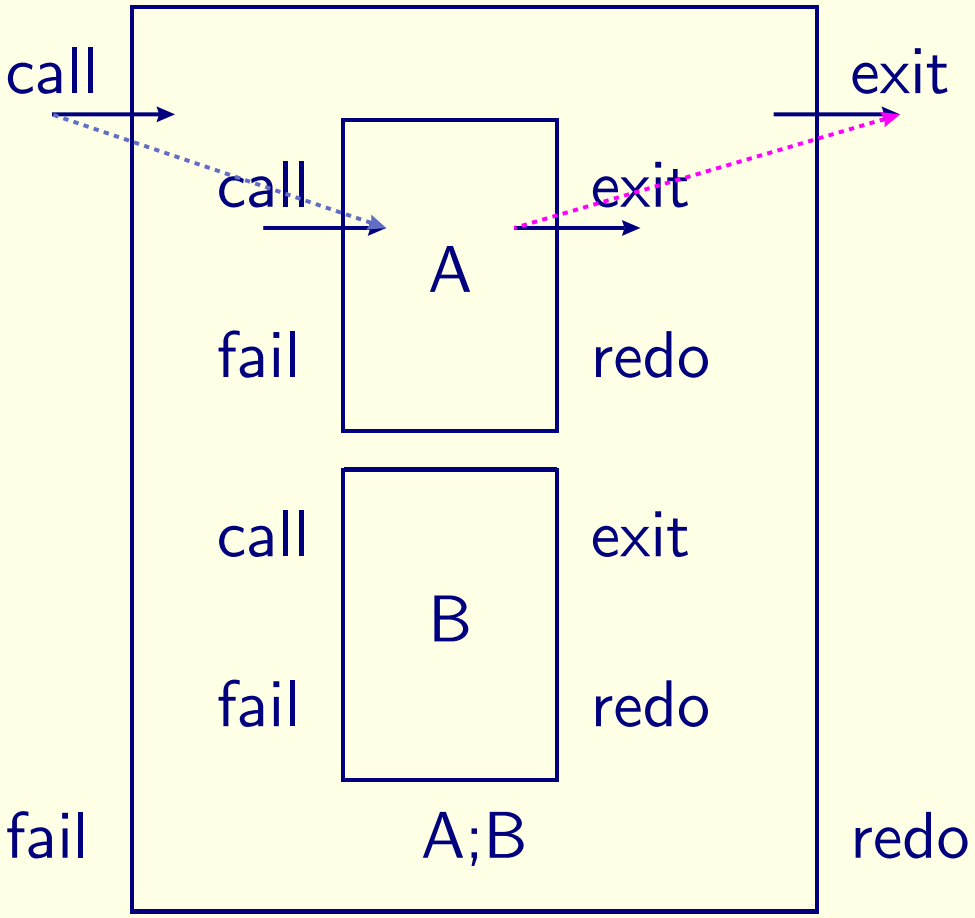
call A;B ⇀ call A

Main idea (cont'd), Disjunction



call A;B ⇀ call A

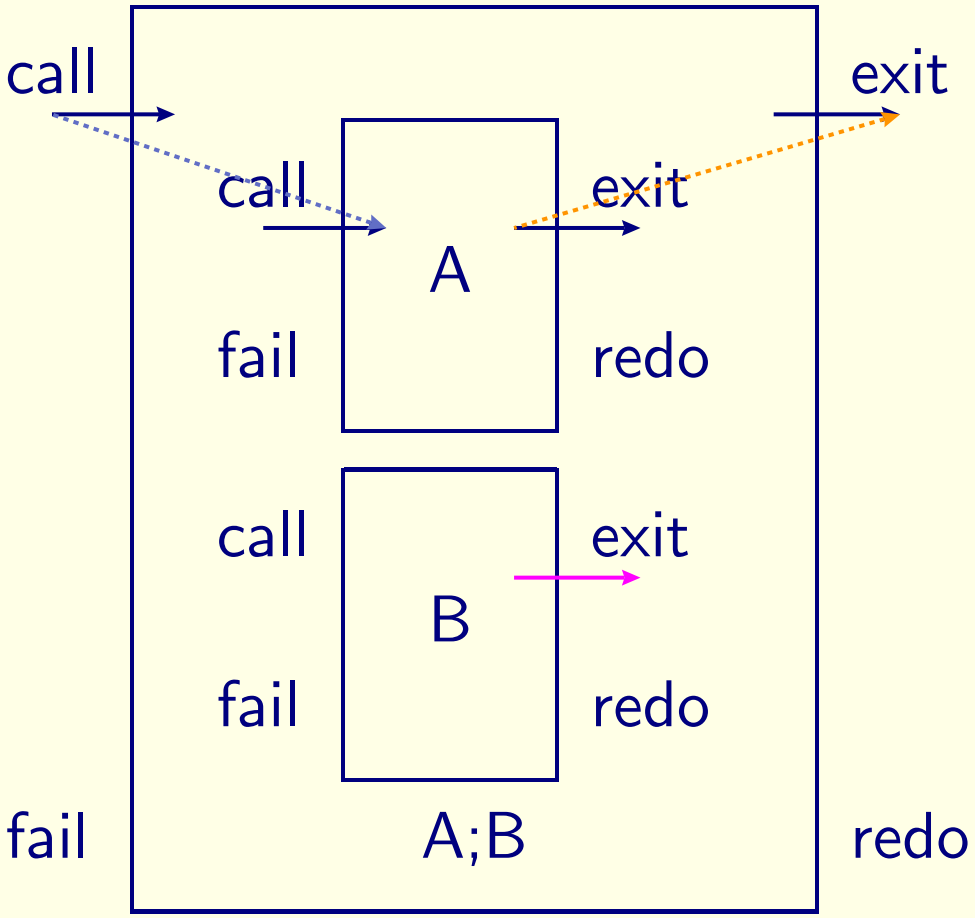
Main idea (cont'd), Disjunction



call A;B $\rightarrow$ call A

exit A $\rightarrow^{\alpha}$ exit A;B

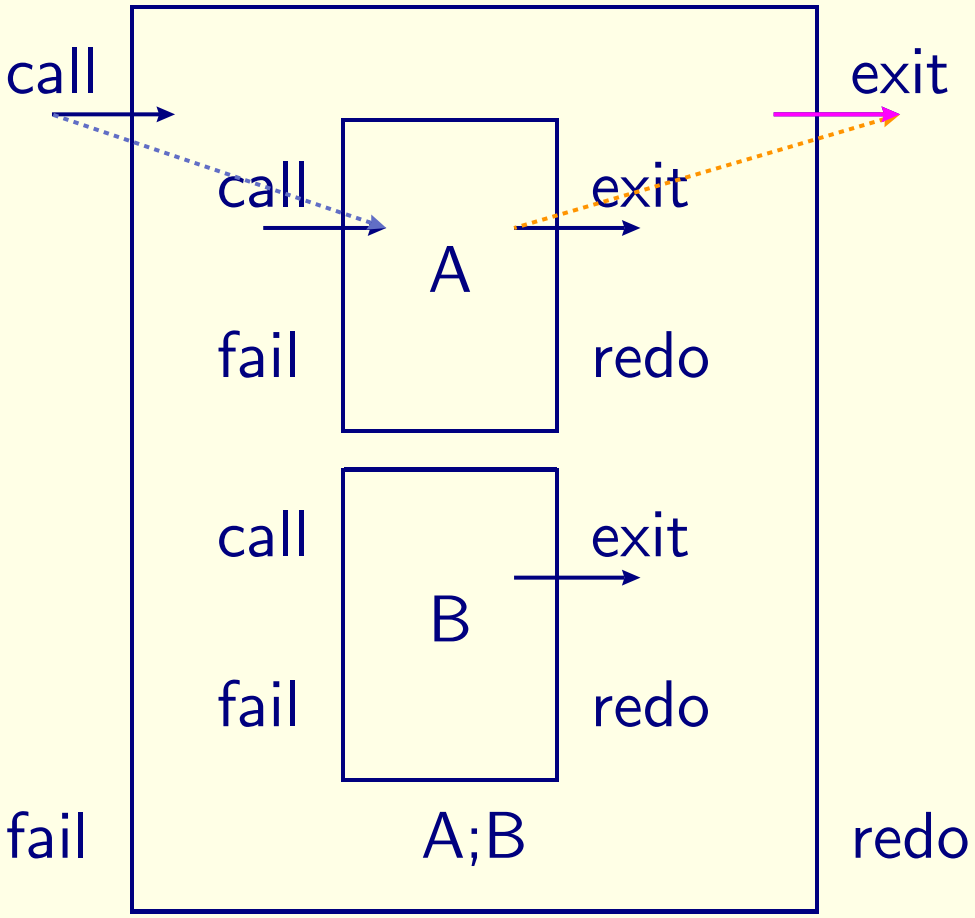
Main idea (cont'd), Disjunction



call A;B $\rightarrowtriangle$ call A

exit A $\rightarrowtriangle^{\alpha}$ exit A;B

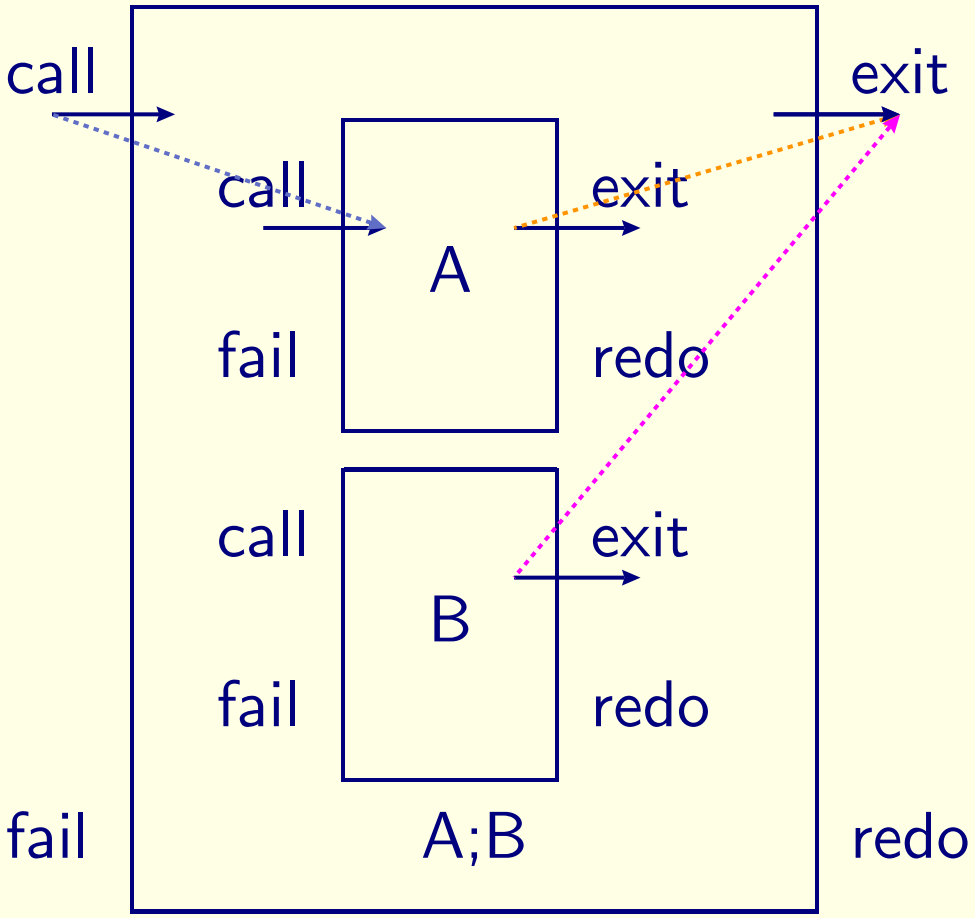
Main idea (cont'd), Disjunction



call A;B $\rightarrowtriangle$ call A

exit A $\rightarrowtriangle^{\alpha}$ exit A;B

Main idea (cont'd), Disjunction

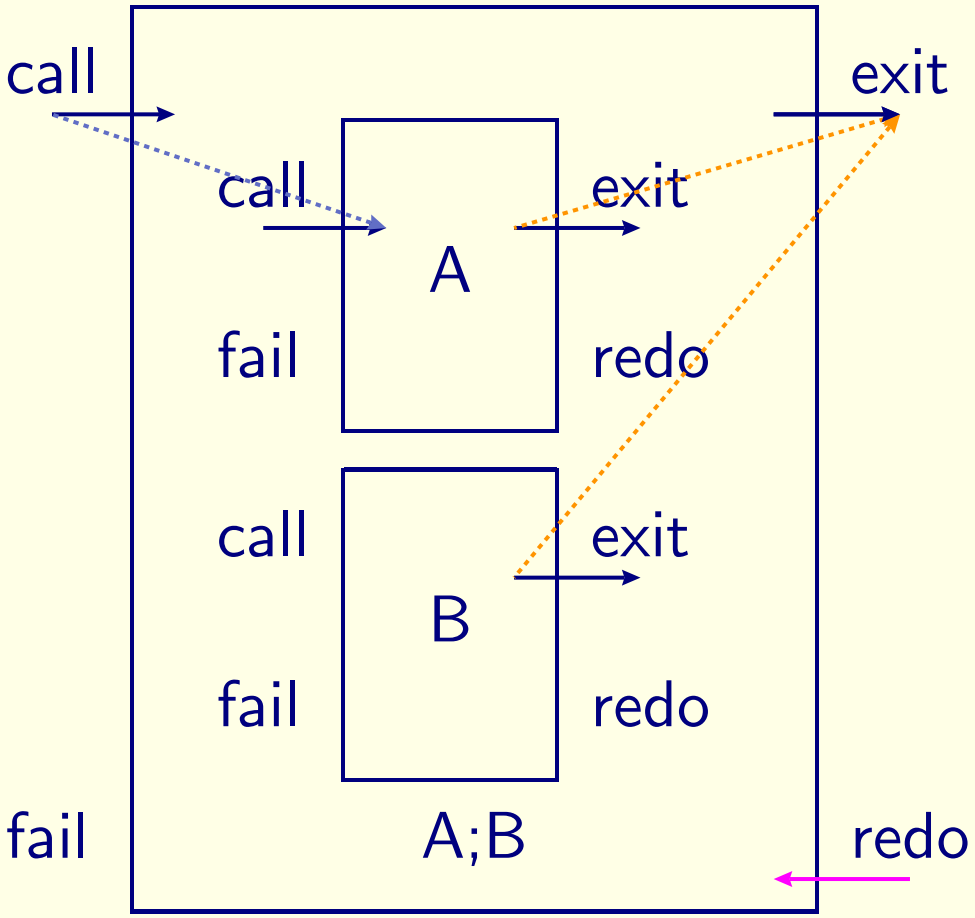


call A;B $\rightarrow$ call A

exit A $\rightarrow^{\alpha}$ exit A;B

exit B $\rightarrow^{\beta}$ exit A;B

Main idea (cont'd), Disjunction

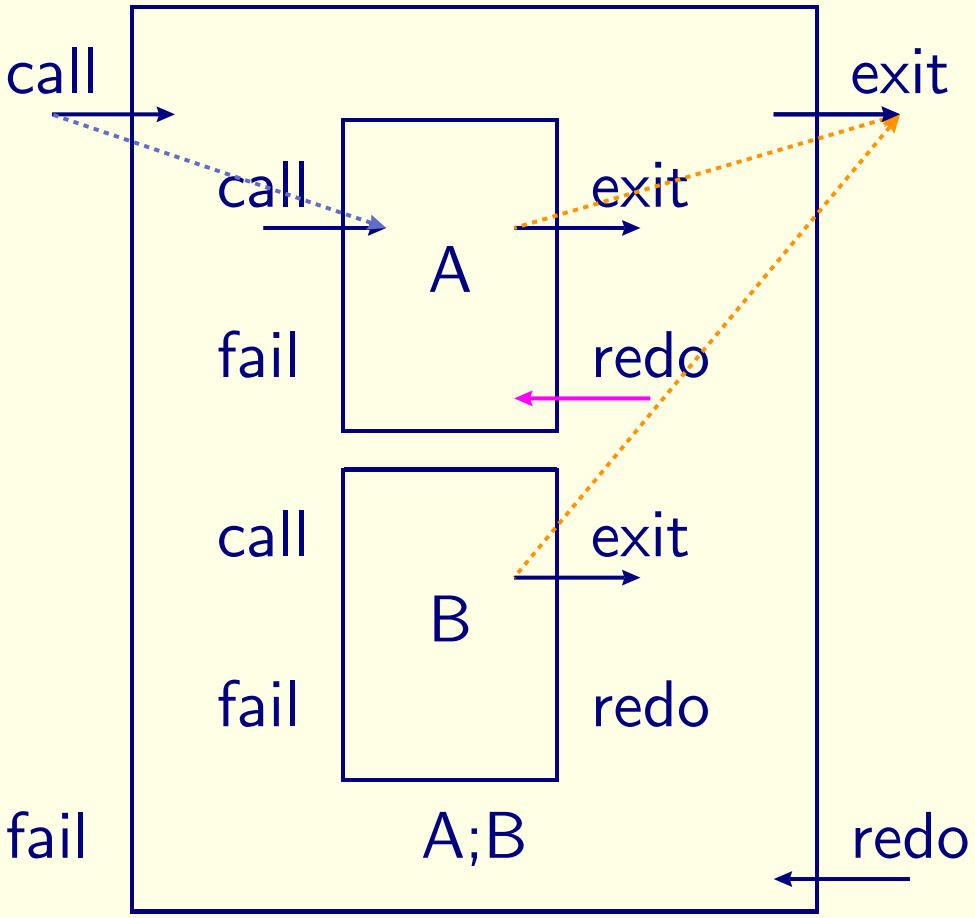


call A;B $\rightarrow$ call A

exit A $\rightarrow^{\alpha}$ exit A;B

exit B $\rightarrow^{\beta}$ exit A;B

Main idea (cont'd), Disjunction

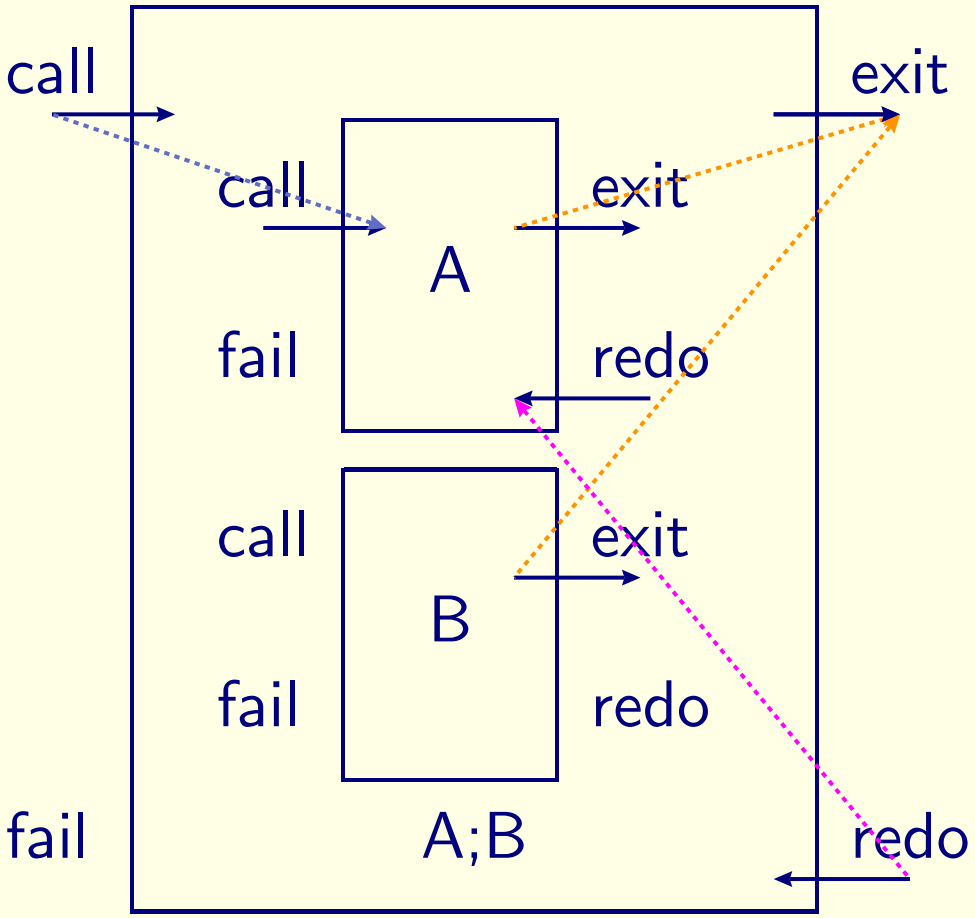


call A;B $\rightarrow$ call A

exit A $\rightarrow^{\alpha}$ exit A;B

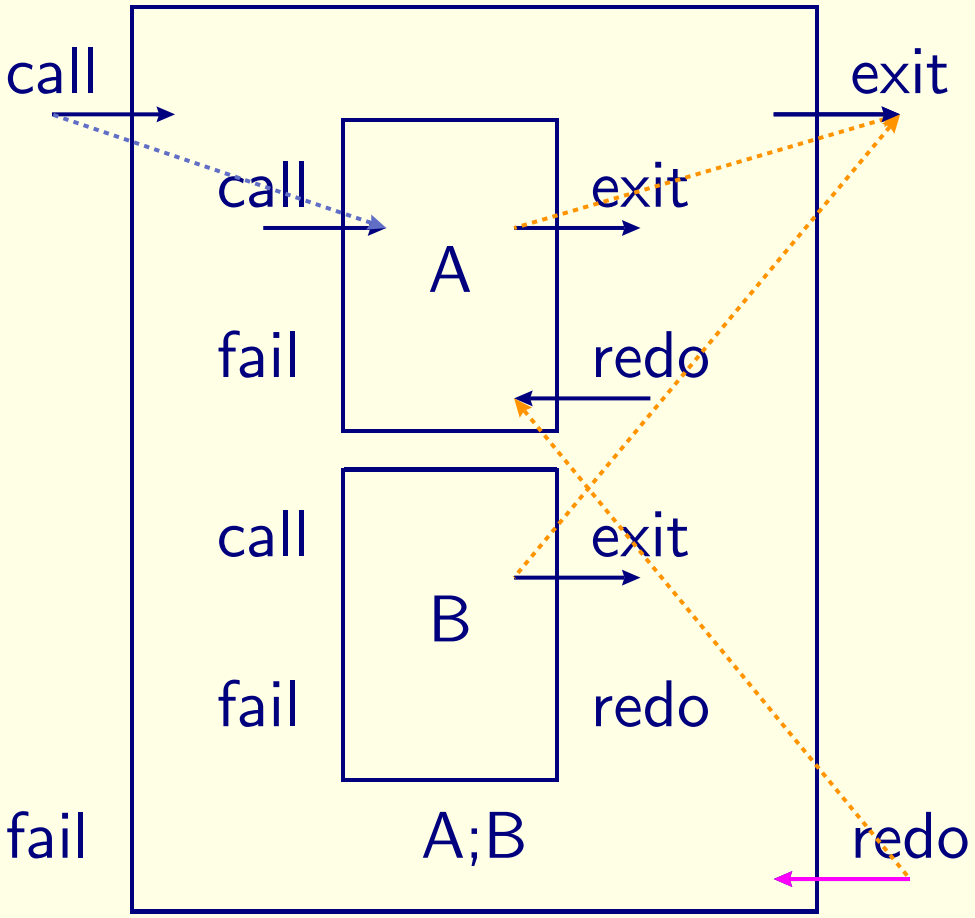
exit B $\rightarrow^{\beta}$ exit A;B

Main idea (cont'd), Disjunction



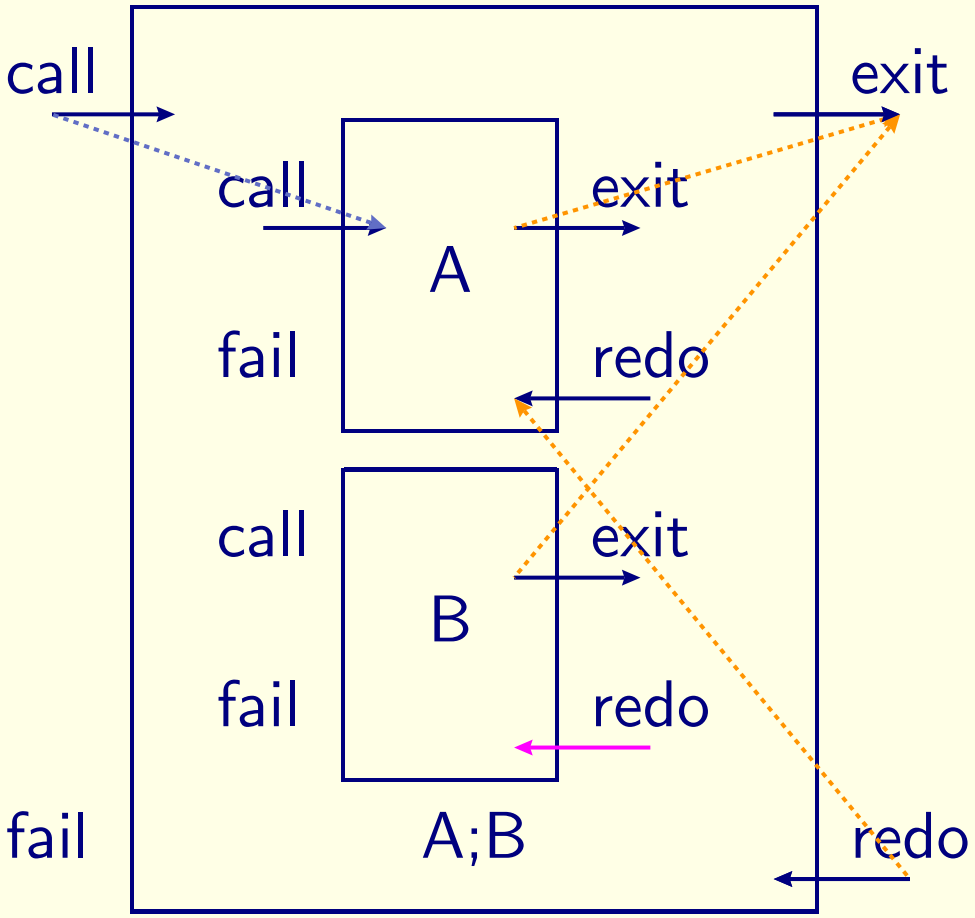
call A;B	$\rightarrowtriangle$	call A
exit A	$\rightarrowtriangle^{\alpha}$	exit A;B
exit B	$\rightarrowtriangle^{\beta}$	exit A;B
$^{\alpha}$redo A;B	$\rightarrowtriangle$	redo A

Main idea (cont'd), Disjunction



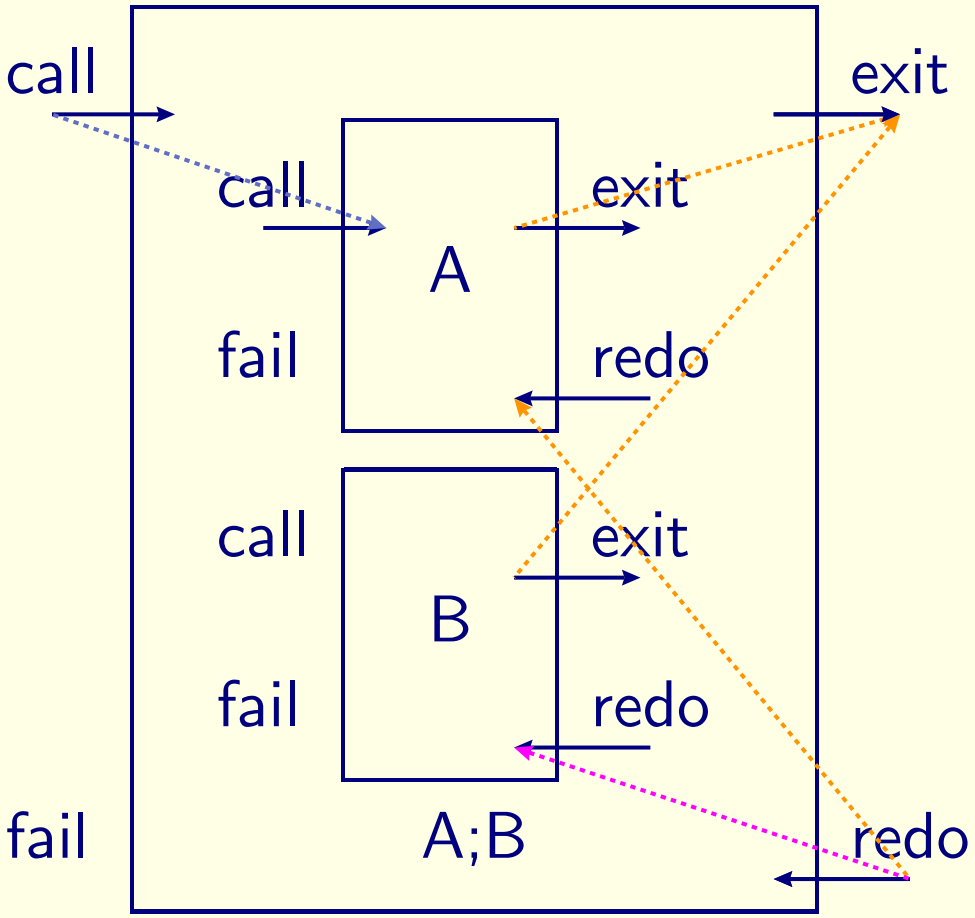
call A;B	$\rightarrow$	call A
exit A	$\rightarrow^{\alpha}$	exit A;B
exit B	$\rightarrow^{\beta}$	exit A;B
$^{\alpha}$redo A;B	$\rightarrow$	redo A

Main idea (cont'd), Disjunction



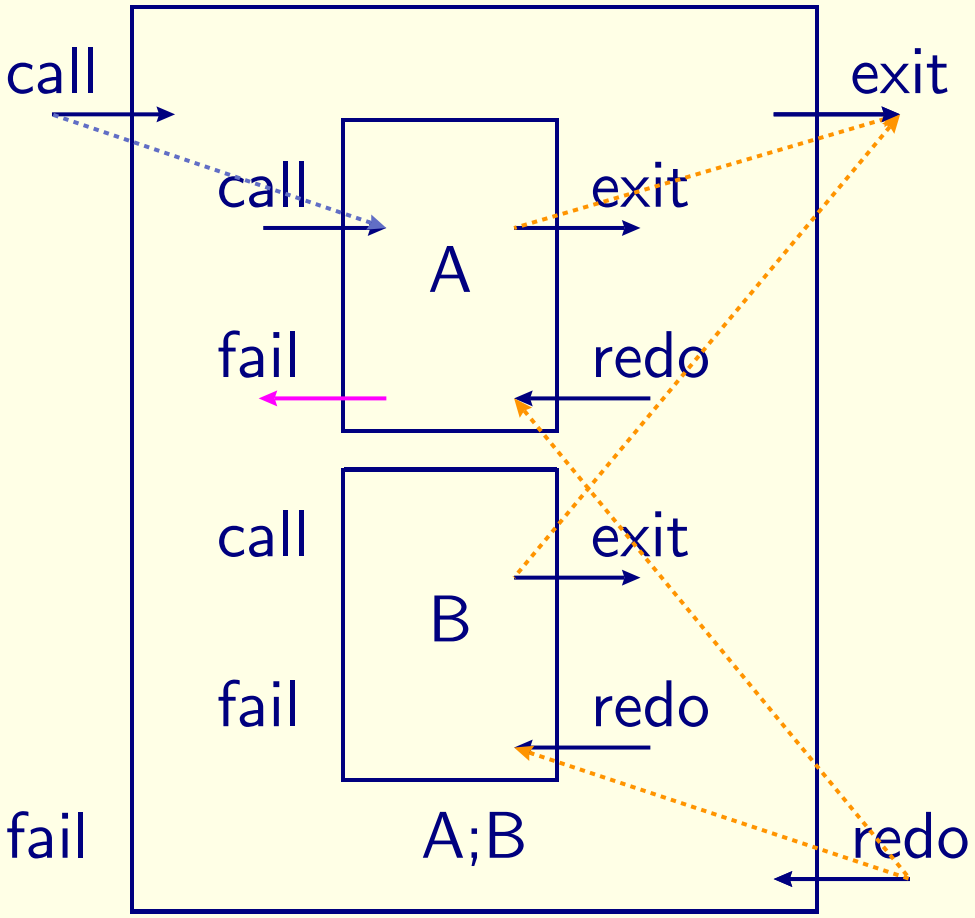
call A;B	$\rightarrow$	call A
exit A	$\rightarrow^{\alpha}$	exit A;B
exit B	$\rightarrow^{\beta}$	exit A;B
$^{\alpha}$redo A;B	$\rightarrow$	redo A

Main idea (cont'd), Disjunction



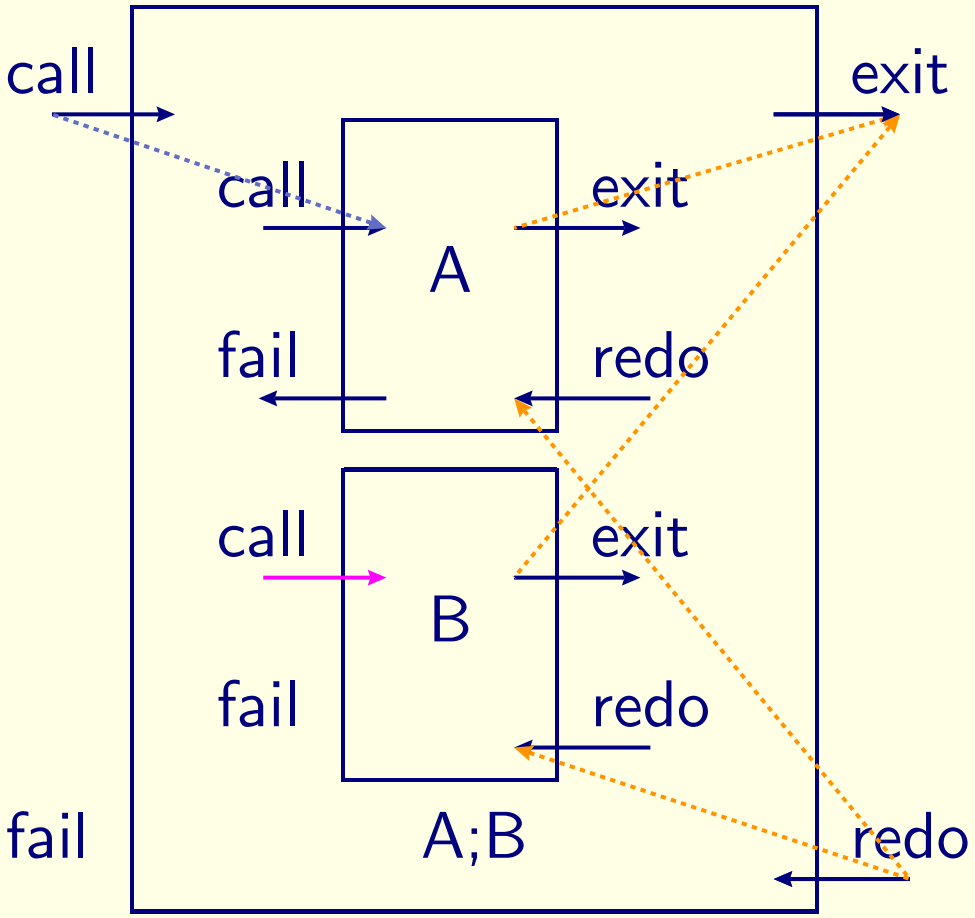
call A;B	$\rightarrow$	call A
exit A	$\rightarrow^{\alpha}$	exit A;B
exit B	$\rightarrow^{\beta}$	exit A;B
$^{\alpha}$redo A;B	$\rightarrow$	redo A
$^{\beta}$redo A;B	$\rightarrow$	redo B

Main idea (cont'd), Disjunction



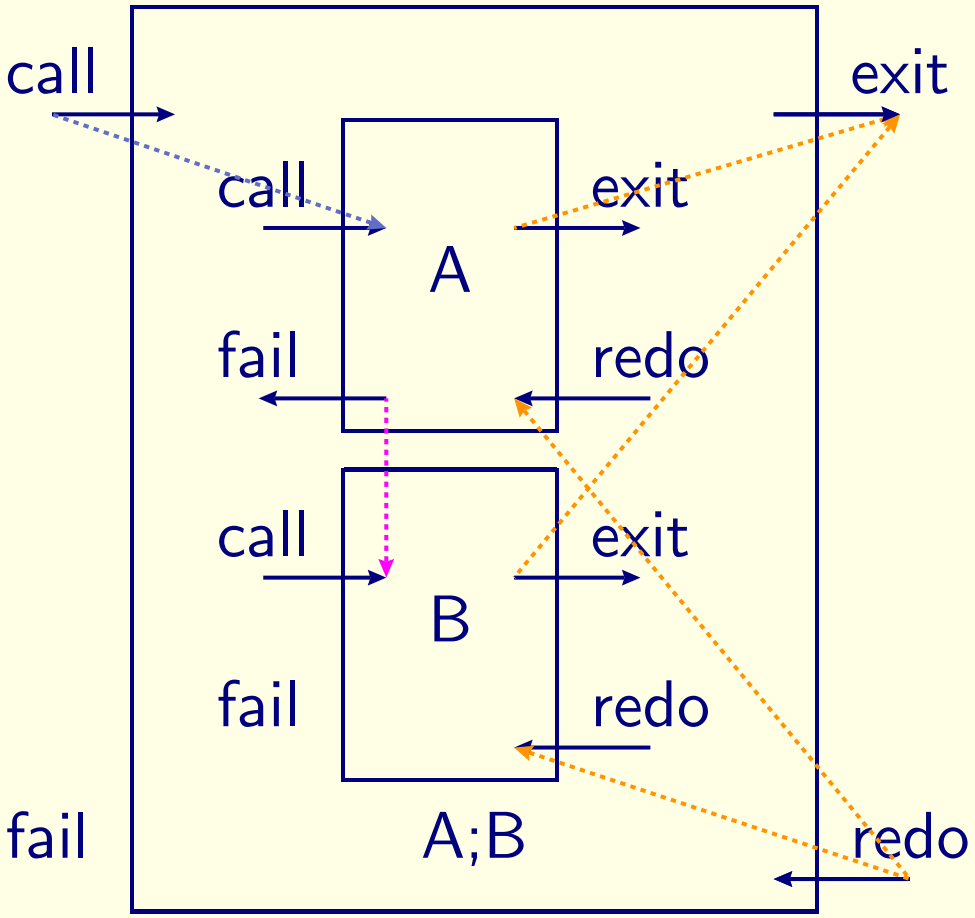
call A;B	$\rightarrow$	call A
exit A	$\rightarrow^{\alpha}$	exit A;B
exit B	$\rightarrow^{\beta}$	exit A;B
$^{\alpha}$redo A;B	$\rightarrow$	redo A
$^{\beta}$redo A;B	$\rightarrow$	redo B

Main idea (cont'd), Disjunction



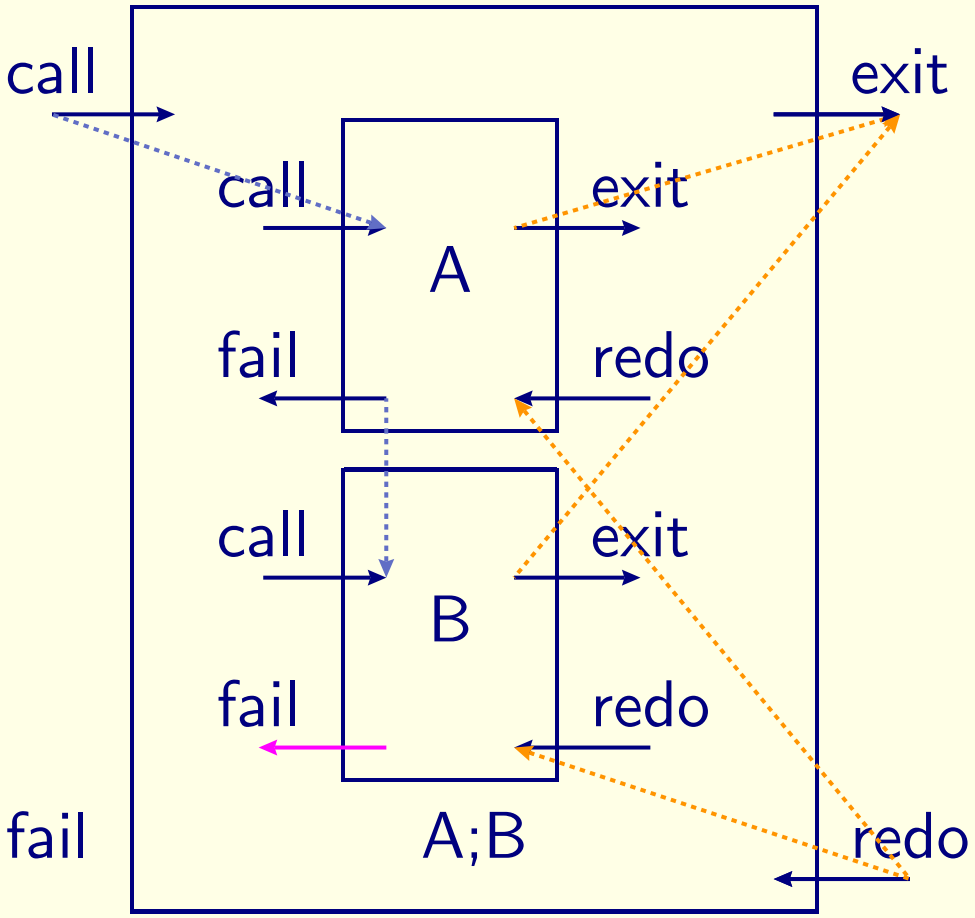
call A;B	$\rightarrow$	call A
exit A	$\rightarrow^{\alpha}$	exit A;B
exit B	$\rightarrow^{\beta}$	exit A;B
$^{\alpha}$redo A;B	$\rightarrow$	redo A
$^{\beta}$redo A;B	$\rightarrow$	redo B

Main idea (cont'd), Disjunction



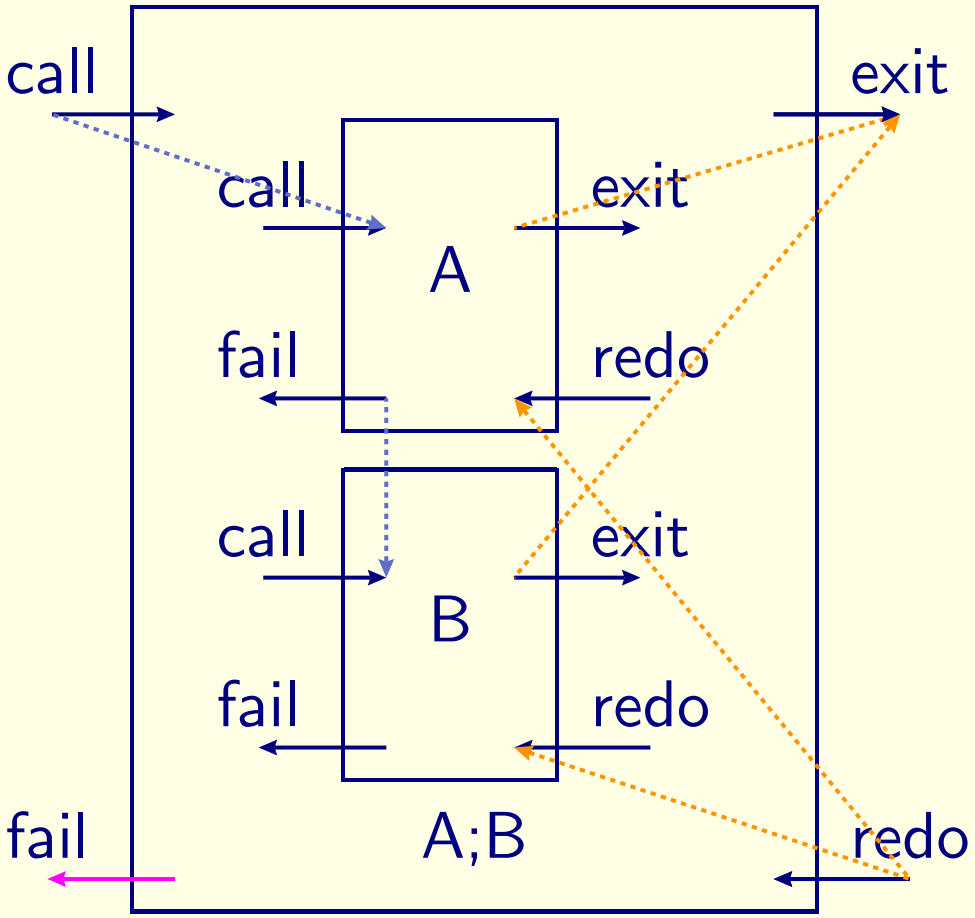
call A;B	$\rightarrow$	call A
exit A	$\rightarrow^{\alpha}$	exit A;B
exit B	$\rightarrow^{\beta}$	exit A;B
$^{\alpha}$redo A;B	$\rightarrow$	redo A
$^{\beta}$redo A;B	$\rightarrow$	redo B
fail A	$\rightarrow$	call B

Main idea (cont'd), Disjunction



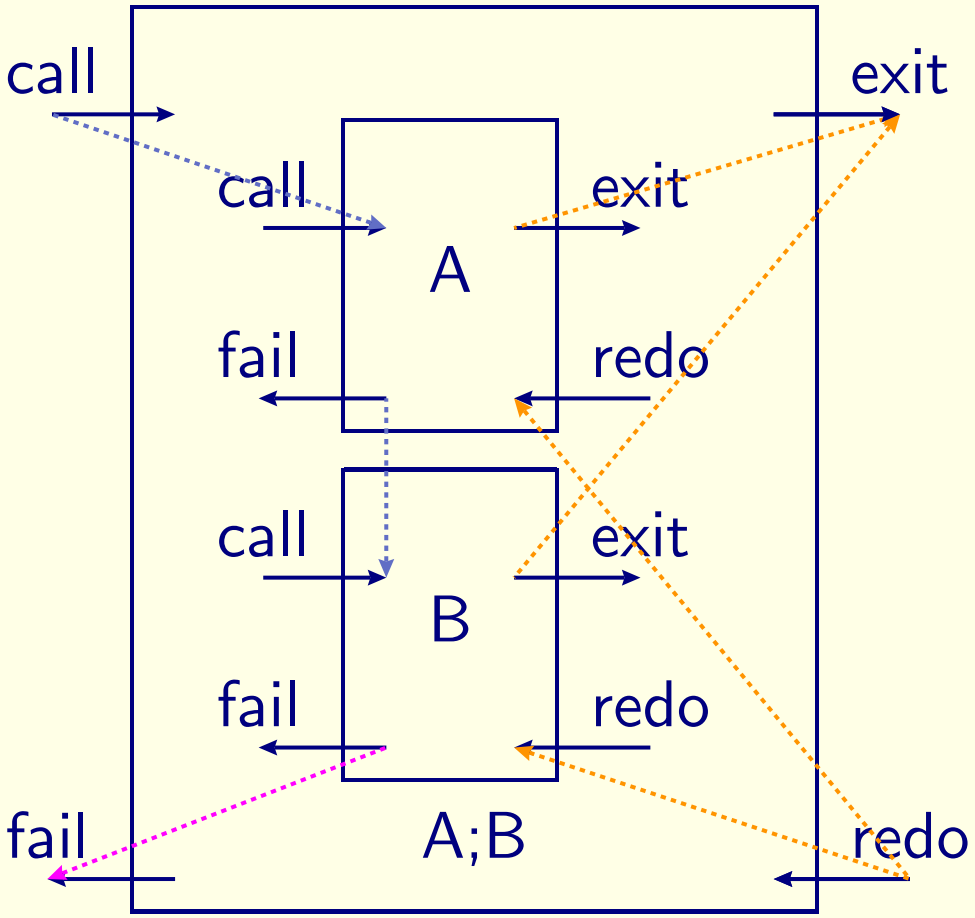
call A;B	$\rightarrow$	call A
exit A	$\rightarrow^{\alpha}$	exit A;B
exit B	$\rightarrow^{\beta}$	exit A;B
$^{\alpha}$redo A;B	$\rightarrow$	redo A
$^{\beta}$redo A;B	$\rightarrow$	redo B
fail A	$\rightarrow$	call B

Main idea (cont'd), Disjunction



call A;B	$\rightarrow$	call A
exit A	$\rightarrow^{\alpha}$	exit A;B
exit B	$\rightarrow^{\beta}$	exit A;B
$^{\alpha}$redo A;B	$\rightarrow$	redo A
$^{\beta}$redo A;B	$\rightarrow$	redo B
fail A	$\rightarrow$	call B

Main idea (cont'd), Disjunction

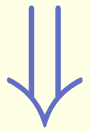


call A;B	$\rightarrow$	call A
exit A	$\rightarrow^{\alpha}$	exit A;B
exit B	$\rightarrow^{\beta}$	exit A;B
$^{\alpha}$redo A;B	$\rightarrow$	redo A
$^{\beta}$redo A;B	$\rightarrow$	redo B
fail A	$\rightarrow$	call B
fail B	$\rightarrow$	fail A;B

A canonical form of Prolog predicates

original program

```
q(a,b).
q(Z,c) :- r(Z).
r(c).
```



canonical representation

```
q(X,Y) :- X=a, Y=b, true; X=Z, Y=c, r(Z).
r(X) :- X=c, true.
```

Filling-in the context

```
main :- good, bad.
good.
```

Filling-in the context

```
main :- good, bad.
good :- true.
```

Filling-in the context

```
main :- good, bad.
good :- true.
```

call main →
  call (good, bad) →
    call good →
      call true →
      exit true →
    exit good →
    call bad →
    fail bad →
    redo good →
      redo true →
      fail true →
    fail good →
  fail (good, bad) →
fail main

Filling-in the context

```
main :- good, bad.
good :- true.
```

call main →
  call (good, bad) →
    call good →
      call true →
      exit true →
    exit good →
    call bad →
    fail bad →
    redo good →
      redo true →
      fail true →
    fail good →
  fail (good, bad) →
fail main

- context information needed: A-stack, stack of ancestors

Filling-in the context

```
main :- good, bad.
good :- true.
```

call main →
  call (good, bad) →
    call good →
      call true →
      exit true →
    exit good →
    call bad →
    fail bad →
    redo good →
      redo true →
      fail true →
    fail good →
  fail (good, bad) →
fail main

- context information needed: A-stack, stack of ancestors
- rest of information: B-stack (bindings for the variables etc.)

```
main :- good, bad.
good :- true.
```

$$call\ \mathsf{main}\,,\ \{nil\},\ \{nil\}$$

```
main :- good, bad.
good :- true.
```

$$call\ \mathsf{main}\ ,\ \{nil\},\ \{nil\}$$

$$\rightarrowtriangle\ \ call\ (\mathsf{good}, \mathsf{bad})\ ,\ \{\mathsf{main} \bullet nil\},\ \{nil\}$$

```
main :- good, bad.
good :- true.
```

$$call\ \mathsf{main}\ ,\ \{nil\},\ \{nil\}$$

$$\rightarrowtriangle\ call\ (\mathsf{good}, \mathsf{bad})\ ,\ \{\mathsf{main} \bullet nil\},\ \{nil\}$$

$$\rightarrowtriangle\ call\ \mathsf{good}\ ,\ \{1/\mathsf{good}, \mathsf{bad} \bullet \mathsf{main} \bullet nil\},\ \{nil\}$$

main :- good, bad.
good :- true.

$$
\begin{aligned}
& call\ \mathsf{main}\ ,\ \{nil\},\ \{nil\} \\
\rightarrowtriangle\ & call\ (\mathsf{good}, \mathsf{bad})\ ,\ \{\mathsf{main} \bullet nil\},\ \{nil\} \\
\rightarrowtriangle\ & call\ \mathsf{good}\ ,\ \{1/\mathsf{good}, \mathsf{bad} \bullet \mathsf{main} \bullet nil\},\ \{nil\} \\
\rightarrowtriangle\ & call\ \mathsf{true}\ ,\ \{\mathsf{good} \bullet 1/\mathsf{good}, \mathsf{bad} \bullet \mathsf{main} \bullet nil\},\ \{nil\}
\end{aligned}
$$

```
main :- good, bad.
good :- true.
```

$$
\begin{aligned}
& call\ \mathsf{main}\ ,\ \{nil\},\ \{nil\} \\
\rightarrowtriangle\ & call\ (\mathsf{good}, \mathsf{bad})\ ,\ \{\mathsf{main} \bullet nil\},\ \{nil\} \\
\rightarrowtriangle\ & call\ \mathsf{good}\ ,\ \{1/\mathsf{good}, \mathsf{bad} \bullet \mathsf{main} \bullet nil\},\ \{nil\} \\
\rightarrowtriangle\ & call\ \mathsf{true}\ ,\ \{\mathsf{good} \bullet 1/\mathsf{good}, \mathsf{bad} \bullet \mathsf{main} \bullet nil\},\ \{nil\} \\
\rightarrowtriangle\ & exit\ \mathsf{true}\ ,\ \{\mathsf{good} \bullet 1/\mathsf{good}, \mathsf{bad} \bullet \mathsf{main} \bullet nil\},\ \{nil\}
\end{aligned}
$$

main :- good, bad.
good :- true.

$$
\begin{aligned}
& call\ \mathsf{main}\ , \{nil\}, \{nil\} \\
\rightarrowtriangle\ & call\ (\mathsf{good}, \mathsf{bad})\ , \{\mathsf{main} \bullet nil\}, \{nil\} \\
\rightarrowtriangle\ & call\ \mathsf{good}\ , \{1/\mathsf{good}, \mathsf{bad} \bullet \mathsf{main} \bullet nil\}, \{nil\} \\
\rightarrowtriangle\ & call\ \mathsf{true}\ , \{\mathsf{good} \bullet 1/\mathsf{good}, \mathsf{bad} \bullet \mathsf{main} \bullet nil\}, \{nil\} \\
\rightarrowtriangle\ & exit\ \mathsf{true}\ , \{\mathsf{good} \bullet 1/\mathsf{good}, \mathsf{bad} \bullet \mathsf{main} \bullet nil\}, \{nil\} \\
\rightarrowtriangle\ & exit\ \mathsf{good}\ , \{1/\mathsf{good}, \mathsf{bad} \bullet \mathsf{main} \bullet nil\}, \{BY(\mathsf{true}) \bullet nil\}
\end{aligned}
$$

```
main :- good, bad.
good :- true.
```

$$
\begin{aligned}
& call\ \mathsf{main}\ ,\ \{nil\},\ \{nil\} \\
\rightarrowtriangle\ & call\ (\mathsf{good}, \mathsf{bad})\ ,\ \{\mathsf{main} \bullet nil\},\ \{nil\} \\
\rightarrowtriangle\ & call\ \mathsf{good}\ ,\ \{1/\mathsf{good}, \mathsf{bad} \bullet \mathsf{main} \bullet nil\},\ \{nil\} \\
\rightarrowtriangle\ & call\ \mathsf{true}\ ,\ \{\mathsf{good} \bullet 1/\mathsf{good}, \mathsf{bad} \bullet \mathsf{main} \bullet nil\},\ \{nil\} \\
\rightarrowtriangle\ & exit\ \mathsf{true}\ ,\ \{\mathsf{good} \bullet 1/\mathsf{good}, \mathsf{bad} \bullet \mathsf{main} \bullet nil\},\ \{nil\} \\
\rightarrowtriangle\ & exit\ \mathsf{good}\ ,\ \{1/\mathsf{good}, \mathsf{bad} \bullet \mathsf{main} \bullet nil\},\ \{BY(\mathsf{true}) \bullet nil\} \\
\rightarrowtriangle\ & call\ \mathsf{bad}\ ,\ \{2/\mathsf{good}, \mathsf{bad} \bullet \mathsf{main} \bullet nil\},\ \{BY(\mathsf{true}) \bullet nil\}
\end{aligned}
$$

main :- good, bad.
good :- true.

$$
\begin{aligned}
& \mathit{call}\ \mathsf{main}\ , \{\mathit{nil}\}, \{\mathit{nil}\} \\
\rightarrowtriangle\ & \mathit{call}\ (\mathsf{good}, \mathsf{bad})\ , \{\mathsf{main} \bullet \mathit{nil}\}, \{\mathit{nil}\} \\
\rightarrowtriangle\ & \mathit{call}\ \mathsf{good}\ , \{\mathit{1}/\mathsf{good}, \mathsf{bad} \bullet \mathsf{main} \bullet \mathit{nil}\}, \{\mathit{nil}\} \\
\rightarrowtriangle\ & \mathit{call}\ \mathsf{true}\ , \{\mathsf{good} \bullet \mathit{1}/\mathsf{good}, \mathsf{bad} \bullet \mathsf{main} \bullet \mathit{nil}\}, \{\mathit{nil}\} \\
\rightarrowtriangle\ & \mathit{exit}\ \mathsf{true}\ , \{\mathsf{good} \bullet \mathit{1}/\mathsf{good}, \mathsf{bad} \bullet \mathsf{main} \bullet \mathit{nil}\}, \{\mathit{nil}\} \\
\rightarrowtriangle\ & \mathit{exit}\ \mathsf{good}\ , \{\mathit{1}/\mathsf{good}, \mathsf{bad} \bullet \mathsf{main} \bullet \mathit{nil}\}, \{\mathit{BY}(\mathsf{true}) \bullet \mathit{nil}\} \\
\rightarrowtriangle\ & \mathit{call}\ \mathsf{bad}\ , \{\mathit{2}/\mathsf{good}, \mathsf{bad} \bullet \mathsf{main} \bullet \mathit{nil}\}, \{\mathit{BY}(\mathsf{true}) \bullet \mathit{nil}\} \\
\rightarrowtriangle\ & \mathit{fail}\ \mathsf{bad}\ , \{\mathit{2}/\mathsf{good}, \mathsf{bad} \bullet \mathsf{main} \bullet \mathit{nil}\}, \{\mathit{BY}(\mathsf{true}) \bullet \mathit{nil}\}
\end{aligned}
$$

```
main :- good, bad.
good :- true.
```

$$
\begin{aligned}
& call\ \mathsf{main}\ ,\ \{nil\},\ \{nil\} \\
\rightarrowtriangle\ & call\ (\mathsf{good}, \mathsf{bad})\ ,\ \{\mathsf{main} \bullet nil\},\ \{nil\} \\
\rightarrowtriangle\ & call\ \mathsf{good}\ ,\ \{1/\mathsf{good}, \mathsf{bad} \bullet \mathsf{main} \bullet nil\},\ \{nil\} \\
\rightarrowtriangle\ & call\ \mathsf{true}\ ,\ \{\mathsf{good} \bullet 1/\mathsf{good}, \mathsf{bad} \bullet \mathsf{main} \bullet nil\},\ \{nil\} \\
\rightarrowtriangle\ & exit\ \mathsf{true}\ ,\ \{\mathsf{good} \bullet 1/\mathsf{good}, \mathsf{bad} \bullet \mathsf{main} \bullet nil\},\ \{nil\} \\
\rightarrowtriangle\ & exit\ \mathsf{good}\ ,\ \{1/\mathsf{good}, \mathsf{bad} \bullet \mathsf{main} \bullet nil\},\ \{BY(\mathsf{true}) \bullet nil\} \\
\rightarrowtriangle\ & call\ \mathsf{bad}\ ,\ \{2/\mathsf{good}, \mathsf{bad} \bullet \mathsf{main} \bullet nil\},\ \{BY(\mathsf{true}) \bullet nil\} \\
\rightarrowtriangle\ & fail\ \mathsf{bad}\ ,\ \{2/\mathsf{good}, \mathsf{bad} \bullet \mathsf{main} \bullet nil\},\ \{BY(\mathsf{true}) \bullet nil\} \\
\rightarrowtriangle\ & redo\ \mathsf{good}\ ,\ \{1/\mathsf{good}, \mathsf{bad} \bullet \mathsf{main} \bullet nil\},\ \{BY(\mathsf{true}) \bullet nil\}
\end{aligned}
$$

main :- good, bad.
good :- true.

$$
\begin{aligned}
& \mathit{call}\ \mathsf{main}\ , \{\mathit{nil}\}, \{\mathit{nil}\} \\
\rightarrowtriangle\ & \mathit{call}\ (\mathsf{good}, \mathsf{bad})\ , \{\mathsf{main} \bullet \mathit{nil}\}, \{\mathit{nil}\} \\
\rightarrowtriangle\ & \mathit{call}\ \mathsf{good}\ , \{\mathit{1}/\mathsf{good}, \mathsf{bad} \bullet \mathsf{main} \bullet \mathit{nil}\}, \{\mathit{nil}\} \\
\rightarrowtriangle\ & \mathit{call}\ \mathsf{true}\ , \{\mathsf{good} \bullet \mathit{1}/\mathsf{good}, \mathsf{bad} \bullet \mathsf{main} \bullet \mathit{nil}\}, \{\mathit{nil}\} \\
\rightarrowtriangle\ & \mathit{exit}\ \mathsf{true}\ , \{\mathsf{good} \bullet \mathit{1}/\mathsf{good}, \mathsf{bad} \bullet \mathsf{main} \bullet \mathit{nil}\}, \{\mathit{nil}\} \\
\rightarrowtriangle\ & \mathit{exit}\ \mathsf{good}\ , \{\mathit{1}/\mathsf{good}, \mathsf{bad} \bullet \mathsf{main} \bullet \mathit{nil}\}, \{\mathit{BY}(\mathsf{true}) \bullet \mathit{nil}\} \\
\rightarrowtriangle\ & \mathit{call}\ \mathsf{bad}\ , \{\mathit{2}/\mathsf{good}, \mathsf{bad} \bullet \mathsf{main} \bullet \mathit{nil}\}, \{\mathit{BY}(\mathsf{true}) \bullet \mathit{nil}\} \\
\rightarrowtriangle\ & \mathit{fail}\ \mathsf{bad}\ , \{\mathit{2}/\mathsf{good}, \mathsf{bad} \bullet \mathsf{main} \bullet \mathit{nil}\}, \{\mathit{BY}(\mathsf{true}) \bullet \mathit{nil}\} \\
\rightarrowtriangle\ & \mathit{redo}\ \mathsf{good}\ , \{\mathit{1}/\mathsf{good}, \mathsf{bad} \bullet \mathsf{main} \bullet \mathit{nil}\}, \{\mathit{BY}(\mathsf{true}) \bullet \mathit{nil}\} \\
\rightarrowtriangle\ & \mathit{redo}\ \mathsf{true}\ , \{\mathsf{good} \bullet \mathit{1}/\mathsf{good}, \mathsf{bad} \bullet \mathsf{main} \bullet \mathit{nil}\}, \{\mathit{nil}\}
\end{aligned}
$$

```
main :- good, bad.
good :- true.
```

$$
\begin{aligned}
& \mathit{call}\ \mathsf{main}\ ,\ \{\mathit{nil}\},\ \{\mathit{nil}\} \\
\rightarrowtriangle\ & \mathit{call}\ (\mathsf{good}, \mathsf{bad})\ ,\ \{\mathsf{main} \bullet \mathit{nil}\},\ \{\mathit{nil}\} \\
\rightarrowtriangle\ & \mathit{call}\ \mathsf{good}\ ,\ \{\mathit{1}/\mathsf{good}, \mathsf{bad} \bullet \mathsf{main} \bullet \mathit{nil}\},\ \{\mathit{nil}\} \\
\rightarrowtriangle\ & \mathit{call}\ \mathsf{true}\ ,\ \{\mathsf{good} \bullet \mathit{1}/\mathsf{good}, \mathsf{bad} \bullet \mathsf{main} \bullet \mathit{nil}\},\ \{\mathit{nil}\} \\
\rightarrowtriangle\ & \mathit{exit}\ \mathsf{true}\ ,\ \{\mathsf{good} \bullet \mathit{1}/\mathsf{good}, \mathsf{bad} \bullet \mathsf{main} \bullet \mathit{nil}\},\ \{\mathit{nil}\} \\
\rightarrowtriangle\ & \mathit{exit}\ \mathsf{good}\ ,\ \{\mathit{1}/\mathsf{good}, \mathsf{bad} \bullet \mathsf{main} \bullet \mathit{nil}\},\ \{BY(\mathsf{true}) \bullet \mathit{nil}\} \\
\rightarrowtriangle\ & \mathit{call}\ \mathsf{bad}\ ,\ \{\mathit{2}/\mathsf{good}, \mathsf{bad} \bullet \mathsf{main} \bullet \mathit{nil}\},\ \{BY(\mathsf{true}) \bullet \mathit{nil}\} \\
\rightarrowtriangle\ & \mathit{fail}\ \mathsf{bad}\ ,\ \{\mathit{2}/\mathsf{good}, \mathsf{bad} \bullet \mathsf{main} \bullet \mathit{nil}\},\ \{BY(\mathsf{true}) \bullet \mathit{nil}\} \\
\rightarrowtriangle\ & \mathit{redo}\ \mathsf{good}\ ,\ \{\mathit{1}/\mathsf{good}, \mathsf{bad} \bullet \mathsf{main} \bullet \mathit{nil}\},\ \{BY(\mathsf{true}) \bullet \mathit{nil}\} \\
\rightarrowtriangle\ & \mathit{redo}\ \mathsf{true}\ ,\ \{\mathsf{good} \bullet \mathit{1}/\mathsf{good}, \mathsf{bad} \bullet \mathsf{main} \bullet \mathit{nil}\},\ \{\mathit{nil}\} \\
\rightarrowtriangle\ & \mathit{fail}\ \mathsf{true}\ ,\ \{\mathsf{good} \bullet \mathit{1}/\mathsf{good}, \mathsf{bad} \bullet \mathsf{main} \bullet \mathit{nil}\},\ \{\mathit{nil}\}
\end{aligned}
$$

```
main :- good, bad.
good :- true.
```

$$
\begin{aligned}
& call\ \mathsf{main}\ , \{nil\}, \{nil\} \\
\rightarrowtriangle\ & call\ (\mathsf{good}, \mathsf{bad})\ , \{\mathsf{main} \bullet nil\}, \{nil\} \\
\rightarrowtriangle\ & call\ \mathsf{good}\ , \{1/\mathsf{good}, \mathsf{bad} \bullet \mathsf{main} \bullet nil\}, \{nil\} \\
\rightarrowtriangle\ & call\ \mathsf{true}\ , \{\mathsf{good} \bullet 1/\mathsf{good}, \mathsf{bad} \bullet \mathsf{main} \bullet nil\}, \{nil\} \\
\rightarrowtriangle\ & exit\ \mathsf{true}\ , \{\mathsf{good} \bullet 1/\mathsf{good}, \mathsf{bad} \bullet \mathsf{main} \bullet nil\}, \{nil\} \\
\rightarrowtriangle\ & exit\ \mathsf{good}\ , \{1/\mathsf{good}, \mathsf{bad} \bullet \mathsf{main} \bullet nil\}, \{BY(\mathsf{true}) \bullet nil\} \\
\rightarrowtriangle\ & call\ \mathsf{bad}\ , \{2/\mathsf{good}, \mathsf{bad} \bullet \mathsf{main} \bullet nil\}, \{BY(\mathsf{true}) \bullet nil\} \\
\rightarrowtriangle\ & fail\ \mathsf{bad}\ , \{2/\mathsf{good}, \mathsf{bad} \bullet \mathsf{main} \bullet nil\}, \{BY(\mathsf{true}) \bullet nil\} \\
\rightarrowtriangle\ & redo\ \mathsf{good}\ , \{1/\mathsf{good}, \mathsf{bad} \bullet \mathsf{main} \bullet nil\}, \{BY(\mathsf{true}) \bullet nil\} \\
\rightarrowtriangle\ & redo\ \mathsf{true}\ , \{\mathsf{good} \bullet 1/\mathsf{good}, \mathsf{bad} \bullet \mathsf{main} \bullet nil\}, \{nil\} \\
\rightarrowtriangle\ & fail\ \mathsf{true}\ , \{\mathsf{good} \bullet 1/\mathsf{good}, \mathsf{bad} \bullet \mathsf{main} \bullet nil\}, \{nil\} \\
\rightarrowtriangle\ & fail\ \mathsf{good}\ , \{1/\mathsf{good}, \mathsf{bad} \bullet \mathsf{main} \bullet nil\}, \{nil\}
\end{aligned}
$$

main :- good, bad.
good :- true.

$call$ main , $\{nil\}$, $\{nil\}$

$\rightarrowtriangle$ $call$ (good, bad) , $\{\text{main} \bullet nil\}$, $\{nil\}$

$\rightarrowtriangle$ $call$ good , $\{1/\text{good}, \text{bad} \bullet \text{main} \bullet nil\}$, $\{nil\}$

$\rightarrowtriangle$ $call$ true , $\{\text{good} \bullet 1/\text{good}, \text{bad} \bullet \text{main} \bullet nil\}$, $\{nil\}$

$\rightarrowtriangle$ $exit$ true , $\{\text{good} \bullet 1/\text{good}, \text{bad} \bullet \text{main} \bullet nil\}$, $\{nil\}$

$\rightarrowtriangle$ $exit$ good , $\{1/\text{good}, \text{bad} \bullet \text{main} \bullet nil\}$, $\{BY(\text{true}) \bullet nil\}$

$\rightarrowtriangle$ $call$ bad , $\{2/\text{good}, \text{bad} \bullet \text{main} \bullet nil\}$, $\{BY(\text{true}) \bullet nil\}$

$\rightarrowtriangle$ $fail$ bad , $\{2/\text{good}, \text{bad} \bullet \text{main} \bullet nil\}$, $\{BY(\text{true}) \bullet nil\}$

$\rightarrowtriangle$ $redo$ good , $\{1/\text{good}, \text{bad} \bullet \text{main} \bullet nil\}$, $\{BY(\text{true}) \bullet nil\}$

$\rightarrowtriangle$ $redo$ true , $\{\text{good} \bullet 1/\text{good}, \text{bad} \bullet \text{main} \bullet nil\}$, $\{nil\}$

$\rightarrowtriangle$ $fail$ true , $\{\text{good} \bullet 1/\text{good}, \text{bad} \bullet \text{main} \bullet nil\}$, $\{nil\}$

$\rightarrowtriangle$ $fail$ good , $\{1/\text{good}, \text{bad} \bullet \text{main} \bullet nil\}$, $\{nil\}$

$\rightarrowtriangle$ $fail$ (good, bad) , $\{\text{main} \bullet nil\}$, $\{nil\}$

main :- good, bad.
good :- true.

$$
\begin{aligned}
& call\ \mathsf{main}\ , \{nil\}, \{nil\} \\
\rightarrowtriangle\ & call\ (\mathsf{good}, \mathsf{bad})\ , \{\mathsf{main} \bullet nil\}, \{nil\} \\
\rightarrowtriangle\ & call\ \mathsf{good}\ , \{1/\mathsf{good}, \mathsf{bad} \bullet \mathsf{main} \bullet nil\}, \{nil\} \\
\rightarrowtriangle\ & call\ \mathsf{true}\ , \{\mathsf{good} \bullet 1/\mathsf{good}, \mathsf{bad} \bullet \mathsf{main} \bullet nil\}, \{nil\} \\
\rightarrowtriangle\ & exit\ \mathsf{true}\ , \{\mathsf{good} \bullet 1/\mathsf{good}, \mathsf{bad} \bullet \mathsf{main} \bullet nil\}, \{nil\} \\
\rightarrowtriangle\ & exit\ \mathsf{good}\ , \{1/\mathsf{good}, \mathsf{bad} \bullet \mathsf{main} \bullet nil\}, \{BY(\mathsf{true}) \bullet nil\} \\
\rightarrowtriangle\ & call\ \mathsf{bad}\ , \{2/\mathsf{good}, \mathsf{bad} \bullet \mathsf{main} \bullet nil\}, \{BY(\mathsf{true}) \bullet nil\} \\
\rightarrowtriangle\ & fail\ \mathsf{bad}\ , \{2/\mathsf{good}, \mathsf{bad} \bullet \mathsf{main} \bullet nil\}, \{BY(\mathsf{true}) \bullet nil\} \\
\rightarrowtriangle\ & redo\ \mathsf{good}\ , \{1/\mathsf{good}, \mathsf{bad} \bullet \mathsf{main} \bullet nil\}, \{BY(\mathsf{true}) \bullet nil\} \\
\rightarrowtriangle\ & redo\ \mathsf{true}\ , \{\mathsf{good} \bullet 1/\mathsf{good}, \mathsf{bad} \bullet \mathsf{main} \bullet nil\}, \{nil\} \\
\rightarrowtriangle\ & fail\ \mathsf{true}\ , \{\mathsf{good} \bullet 1/\mathsf{good}, \mathsf{bad} \bullet \mathsf{main} \bullet nil\}, \{nil\} \\
\rightarrowtriangle\ & fail\ \mathsf{good}\ , \{1/\mathsf{good}, \mathsf{bad} \bullet \mathsf{main} \bullet nil\}, \{nil\} \\
\rightarrowtriangle\ & fail\ (\mathsf{good}, \mathsf{bad})\ , \{\mathsf{main} \bullet nil\}, \{nil\} \\
\rightarrowtriangle\ & fail\ \mathsf{main}\ , \{nil\}, \{nil\}
\end{aligned}
$$

S_1:PP Language of events

Example $\mathit{fail}\ \mathsf{bad}\ \langle \frac{BY(\mathsf{true}) \bullet \mathit{nil}}{2/\mathsf{good,bad} \bullet \mathsf{main} \bullet \mathit{nil}} \rangle$

Grammar

event	::=	port goal $\langle \frac{\text{B-stack}}{\text{A-stack}} \rangle$
port	::=	$\mathit{call} \mid \mathit{exit} \mid \mathit{fail} \mid \mathit{redo}$
goal	::=	true | fail | atom | term=term | goal;goal | goal,goal
A-stack	::=	nil | ancestor • A-stack
ancestor	::=	atom | tag/goal;goal | tag/goal,goal
tag	::=	$1 \mid 2$
B-stack	::=	nil | bet • B-stack
bet	::=	mgu | memo
memo	::=	$BY(\text{goal}) \mid OR(\text{tag})$

S_1:PP Language of events

Example $fail$ bad $\langle \frac{BY(\text{true}) \bullet nil}{2/\text{good,bad} \bullet \text{main} \bullet nil} \rangle$

Grammar

event	::=	port goal $\langle \frac{\text{B-stack}}{\text{A-stack}} \rangle$
port	::=	$call$ | $exit$ | $fail$ | $redo$
goal	::=	true | fail | atom | term=term | goal;goal | goal,goal
A-stack	::=	nil | ancestor • A-stack
ancestor	::=	atom | tag/goal;goal | tag/goal,goal
tag	::=	1 | 2
B-stack	::=	nil | bet • B-stack
bet	::=	mgu | memo
memo	::=	BY(goal) | OR(tag)

S_1:PP Language of events

Example $fail$ bad $\langle \frac{BY(\text{true}) \bullet nil}{2/\text{good,bad} \bullet \text{main} \bullet nil} \rangle$

Grammar

event	::=	port goal $\langle \frac{\text{B-stack}}{\text{A-stack}} \rangle$
port	::=	$call$ | $exit$ | $fail$ | $redo$
goal	::=	true | fail | atom | term=term | goal;goal | goal,goal
A-stack	::=	nil | ancestor • A-stack
ancestor	::=	atom | tag/goal;goal | tag/goal,goal
tag	::=	1 | 2
B-stack	::=	nil | bet • B-stack
bet	::=	mgu | memo
memo	::=	BY(goal) | OR(tag)

S_1:PP Language of events

Example $fail$ bad $\langle \frac{BY(\text{true}) \bullet nil}{2/\text{good,bad} \bullet \text{main} \bullet nil} \rangle$

Grammar

event	::=	port goal $\langle \frac{\text{B-stack}}{\text{A-stack}} \rangle$
port	::=	$call \mid exit \mid fail \mid redo$
goal	::=	true | fail | atom | term=term | goal;goal | goal,goal
A-stack	::=	nil | ancestor • A-stack
ancestor	::=	atom | tag/goal;goal | tag/goal,goal
tag	::=	$1 \mid 2$
B-stack	::=	nil | bet • B-stack
bet	::=	mgu | memo
memo	::=	$BY(\text{goal}) \mid OR(\text{tag})$

S_1:PP Language of events

Example $fail$ bad $\langle \dfrac{BY(\text{true}) \bullet nil}{2/\text{good,bad} \bullet \text{main} \bullet nil} \rangle$

Grammar

event	::=	port goal $\langle \frac{\text{B-stack}}{\text{A-stack}} \rangle$
port	::=	$call \mid exit \mid fail \mid redo$
goal	::=	true | fail | atom | term=term | goal;goal | goal,goal
A-stack	::=	nil | ancestor • A-stack
ancestor	::=	atom | tag/goal;goal | tag/goal,goal
tag	::=	$1 \mid 2$
B-stack	::=	nil | bet • B-stack
bet	::=	mgu | memo
memo	::=	$BY(\text{goal}) \mid OR(\text{tag})$

S_1:PP Language of events

Example $fail$ bad $\langle \frac{BY(\text{true}) \bullet nil}{2/\text{good,bad} \bullet \text{main} \bullet nil} \rangle$

Grammar

event	::=	port goal $\langle \frac{\text{B-stack}}{\text{A-stack}} \rangle$
port	::=	$call \mid exit \mid fail \mid redo$
goal	::=	true | fail | atom | term=term | goal;goal | goal,goal
A-stack	::=	nil | ancestor • A-stack
ancestor	::=	atom | tag/goal;goal | tag/goal,goal
tag	::=	$1 \mid 2$
B-stack	::=	nil | bet • B-stack
bet	::=	mgu | memo
memo	::=	$BY(\text{goal}) \mid OR(\text{tag})$

S_1:PP Language of events

Example $fail$ bad $\langle \frac{BY(\text{true}) \bullet nil}{2/\text{good,bad} \bullet \text{main} \bullet nil} \rangle$

Grammar

event	::=	port goal $\langle \frac{\text{B-stack}}{\text{A-stack}} \rangle$
port	::=	$call$ | $exit$ | $fail$ | $redo$
goal	::=	true | fail | atom | term=term | goal;goal | goal,goal
A-stack	::=	nil | ancestor • A-stack
ancestor	::=	atom | tag/goal;goal | tag/goal,goal
tag	::=	1 | 2
B-stack	::=	nil | bet • B-stack
bet	::=	mgu | memo
memo	::=	BY(goal) | OR(tag)

S_1:PP Language of events

Example $fail$ bad $\langle \frac{BY(\text{true}) \bullet nil}{2/\text{good,bad} \bullet \text{main} \bullet nil} \rangle$

Grammar

event	::=	port goal $\langle \frac{\text{B-stack}}{\text{A-stack}} \rangle$
port	::=	$call$ | $exit$ | $fail$ | $redo$
goal	::=	true | fail | atom | term=term | goal;goal | goal,goal
A-stack	::=	nil | ancestor • A-stack
ancestor	::=	atom | tag/goal;goal | tag/goal,goal
tag	::=	1 | 2
B-stack	::=	nil | bet • B-stack
bet	::=	mgu | memo
memo	::=	BY(goal) | OR(tag)

S_1:PP Calculus in 20 rules

First of all, how to specify the built-in predicates true and fail?

True

Fail

S_1:PP Calculus in 20 rules

First of all, how to specify the built-in predicates true and fail?

True

$$call\ \text{true}\ \langle \frac{\Sigma}{U} \rangle \rightarrowtriangle exit\ \text{true}\ \langle \frac{\Sigma}{U} \rangle \qquad (S_1\text{:true:1})$$

Fail

S_1:PP Calculus in 20 rules

First of all, how to specify the built-in predicates true and fail?

True

$$call\ \text{true}\ \langle \frac{\Sigma}{U} \rangle \ \rightarrowtriangle\ exit\ \text{true}\ \langle \frac{\Sigma}{U} \rangle \qquad (S_1\text{:true:1})$$

$$redo\ \text{true}\ \langle \frac{\Sigma}{U} \rangle \ \rightarrowtriangle\ fail\ \text{true}\ \langle \frac{\Sigma}{U} \rangle \qquad (S_1\text{:true:2})$$

Fail

S_1:PP Calculus in 20 rules

First of all, how to specify the built-in predicates true and fail?

True

$$call\ \text{true}\ \langle \frac{\Sigma}{U} \rangle \ \rightarrowtriangle\ exit\ \text{true}\ \langle \frac{\Sigma}{U} \rangle \qquad (S_1\text{:true:1})$$

$$redo\ \text{true}\ \langle \frac{\Sigma}{U} \rangle \ \rightarrowtriangle\ fail\ \text{true}\ \langle \frac{\Sigma}{U} \rangle \qquad (S_1\text{:true:2})$$

Fail

$$call\ \text{fail}\ \langle \frac{\Sigma}{U} \rangle \ \rightarrowtriangle\ fail\ \text{fail}\ \langle \frac{\Sigma}{U} \rangle \qquad (S_1\text{:fail})$$

S_1:PP Calculus: **Explicit unification**

(S_1:unif:1)

(S_1:unif:2)

S_1:PP Calculus: **Explicit unification**

$$call\ T_1 = T_2\ \langle \frac{\boldsymbol{\Sigma}}{\boldsymbol{U}} \rangle \rightarrow \begin{cases} exit\ T_1 = T_2\ \langle \frac{\sigma \bullet \boldsymbol{\Sigma}}{\boldsymbol{U}} \rangle, & \text{if } mgu\ldots^{\dagger} \\ fail\ T_1 = T_2\ \langle \frac{\boldsymbol{\Sigma}}{\boldsymbol{U}} \rangle, & \text{otherwise} \end{cases} \qquad (S_1\text{:unif:1})$$

$(S_1\text{:unif:2})$

---

† if $mgu(\boldsymbol{\Sigma}(T_1), \boldsymbol{\Sigma}(T_2)) = \sigma$.

S_1:PP Calculus: **Explicit unification**

$$call\ T_1 = T_2\ \langle \frac{\Sigma}{U} \rangle \ \rightarrow\ \begin{cases} exit\ T_1 = T_2\ \langle \frac{\sigma \bullet \Sigma}{U} \rangle, & \text{if } mgu\ldots^{\dagger} \\ fail\ T_1 = T_2\ \langle \frac{\Sigma}{U} \rangle, & \text{otherwise} \end{cases} \qquad (S_1\text{:unif:1})$$

$$redo\ T_1 = T_2\ \langle \frac{\sigma \bullet \Sigma}{U} \rangle \ \rightarrow\ fail\ T_1 = T_2\ \langle \frac{\Sigma}{U} \rangle \qquad (S_1\text{:unif:2})$$

---

† if $mgu(\Sigma(T_1), \Sigma(T_2)) = \sigma$.

S_1:PP Calculus: User-defined atom

(S_1:atom:1)

(S_1:atom:2)

(S_1:atom:3)

(S_1:atom:4)

S_1:PP Calculus: User-defined atom

$$call\ G_A\ \langle \frac{\Sigma}{U} \rangle \rightarrowtriangle \begin{cases} call\ \sigma(B)\ \langle \frac{\Sigma}{G_A \bullet U} \rangle, & \text{if } H{:-}B\ \ldots^{\ddagger} \\ fail\ G_A\ \langle \frac{\Sigma}{U} \rangle, & \text{otherwise} \end{cases} \qquad (S_1\text{:atom:1})$$

(S_1:atom:2)

(S_1:atom:3)

(S_1:atom:4)

[‡] if $H{:-}B$ is a fresh renaming of a clause in Π, and $\sigma = mgu(G'_A, H)$ with $G'_A := \Sigma(G_A)$ and $\sigma(G'_A) = G'_A$.

S_1:PP Calculus: User-defined atom

$$call\ G_A\ \langle \frac{\Sigma}{U} \rangle \ \rightarrowtriangle \begin{cases} call\ \sigma(B)\ \langle \frac{\Sigma}{G_A \bullet U} \rangle, & \text{if } H{:-}B\ \ldots^{\ddagger} \\ fail\ G_A\ \langle \frac{\Sigma}{U} \rangle, & \text{otherwise} \end{cases} \qquad (S_1\text{:atom:1})$$

$$exit\ B\ \langle \frac{\Sigma}{G_A \bullet U} \rangle \ \rightarrowtriangle \ exit\ G_A\ \langle \frac{BY(B) \bullet \Sigma}{U} \rangle \qquad (S_1\text{:atom:2})$$

(S_1:atom:3)

(S_1:atom:4)

[‡] if $H{:-}B$ is a fresh renaming of a clause in Π, and $\sigma = mgu(G'_A, H)$ with $G'_A := \Sigma(G_A)$ and $\sigma(G'_A) = G'_A$.

S_1:PP Calculus: User-defined atom

$$call\ G_A\ \langle \frac{\Sigma}{U} \rangle \ \rightarrowtriangle \begin{cases} call\ \sigma(B)\ \langle \frac{\Sigma}{G_A \bullet U} \rangle, & \text{if } H{:-}B\ \ldots^{\ddagger} \\ fail\ G_A\ \langle \frac{\Sigma}{U} \rangle, & \text{otherwise} \end{cases} \qquad (S_1\text{:atom:1})$$

$$exit\ B\ \langle \frac{\Sigma}{G_A \bullet U} \rangle \ \rightarrowtriangle \ exit\ G_A\ \langle \frac{BY(B) \bullet \Sigma}{U} \rangle \qquad (S_1\text{:atom:2})$$

$$fail\ B\ \langle \frac{\Sigma}{G_A \bullet U} \rangle \ \rightarrowtriangle \ fail\ G_A\ \langle \frac{\Sigma}{U} \rangle \qquad (S_1\text{:atom:3})$$

$(S_1\text{:atom:4})$

[‡] if $H{:-}B$ is a fresh renaming of a clause in Π, and $\sigma = mgu(G'_A, H)$ with $G'_A := \Sigma(G_A)$ and $\sigma(G'_A) = G'_A$.

S_1:PP Calculus: User-defined atom

$$call\ G_A\ \langle \frac{\Sigma}{U} \rangle \ \rightarrowtriangle \begin{cases} call\ \sigma(B)\ \langle \frac{\Sigma}{G_A \bullet U} \rangle, & \text{if } H{:-}B\ \ldots^{\ddagger} \\ fail\ G_A\ \langle \frac{\Sigma}{U} \rangle, & \text{otherwise} \end{cases} \qquad (S_1\text{:atom:1})$$

$$exit\ B\ \langle \frac{\Sigma}{G_A \bullet U} \rangle \ \rightarrowtriangle \ exit\ G_A\ \langle \frac{BY(B) \bullet \Sigma}{U} \rangle \qquad (S_1\text{:atom:2})$$

$$fail\ B\ \langle \frac{\Sigma}{G_A \bullet U} \rangle \ \rightarrowtriangle \ fail\ G_A\ \langle \frac{\Sigma}{U} \rangle \qquad (S_1\text{:atom:3})$$

$$redo\ G_A\ \langle \frac{BY(B) \bullet \Sigma}{U} \rangle \ \rightarrowtriangle \ redo\ B\ \langle \frac{\Sigma}{G_A \bullet U} \rangle \qquad (S_1\text{:atom:4})$$

---

‡ if $H{:-}B$ is a fresh renaming of a clause in Π, and $\sigma = mgu(G'_A, H)$ with $G'_A := \Sigma(G_A)$ and $\sigma(G'_A) = G'_A$.

S_1:PP Calculus: **Conjunction**

(S_1:conj:1)

(S_1:conj:2)

(S_1:conj:3)

(S_1:conj:4)

(S_1:conj:5)

(S_1:conj:6)

S_1:PP Calculus: **Conjunction**

$$call\ A, B\ \langle \frac{\Sigma}{U} \rangle \rightarrowtriangle call\ A\ \langle \frac{\Sigma}{1/A,B \bullet U} \rangle \qquad (S_1\text{:conj:1})$$

(S_1:conj:2)

(S_1:conj:3)

(S_1:conj:4)

(S_1:conj:5)

(S_1:conj:6)

S_1:PP Calculus: **Conjunction**

$$call\ A, B\ \langle \frac{\Sigma}{U} \rangle \ \rightarrowtriangle\ call\ A\ \langle \frac{\Sigma}{1/A,B \bullet U} \rangle \qquad (S_1\text{:conj:1})$$

$$exit\ A\ \langle \frac{\Sigma}{1/A,B \bullet U} \rangle \ \rightarrowtriangle\ call\ B\ \langle \frac{\Sigma}{2/A,B \bullet U} \rangle \qquad (S_1\text{:conj:2})$$

(S_1:conj:3)

(S_1:conj:4)

(S_1:conj:5)

(S_1:conj:6)

S_1:PP Calculus: **Conjunction**

$$call\ A, B\ \langle \frac{\Sigma}{U} \rangle \ \rightarrowtriangle\ call\ A\ \langle \frac{\Sigma}{1/A,B \bullet U} \rangle \qquad (S_1\text{:conj:1})$$

$$exit\ A\ \langle \frac{\Sigma}{1/A,B \bullet U} \rangle \ \rightarrowtriangle\ call\ B\ \langle \frac{\Sigma}{2/A,B \bullet U} \rangle \qquad (S_1\text{:conj:2})$$

$$fail\ A\ \langle \frac{\Sigma}{1/A,B \bullet U} \rangle \ \rightarrowtriangle\ fail\ A, B\ \langle \frac{\Sigma}{U} \rangle \qquad (S_1\text{:conj:3})$$

(S$_1$:conj:4)

(S$_1$:conj:5)

(S$_1$:conj:6)

S_1:PP Calculus: **Conjunction**

$$call\ A, B\ \langle \frac{\Sigma}{U} \rangle \rightarrowtriangle call\ A\ \langle \frac{\Sigma}{1/A,B \bullet U} \rangle \quad (S_1\text{:conj:1})$$

$$exit\ A\ \langle \frac{\Sigma}{1/A,B \bullet U} \rangle \rightarrowtriangle call\ B\ \langle \frac{\Sigma}{2/A,B \bullet U} \rangle \quad (S_1\text{:conj:2})$$

$$fail\ A\ \langle \frac{\Sigma}{1/A,B \bullet U} \rangle \rightarrowtriangle fail\ A, B\ \langle \frac{\Sigma}{U} \rangle \quad (S_1\text{:conj:3})$$

$$exit\ B\ \langle \frac{\Sigma}{2/A,B \bullet U} \rangle \rightarrowtriangle exit\ A, B\ \langle \frac{\Sigma}{U} \rangle \quad (S_1\text{:conj:4})$$

(S_1:conj:5)

(S_1:conj:6)

S_1:PP Calculus: **Conjunction**

$$call\ A, B\ \langle \frac{\Sigma}{U} \rangle \rightarrow call\ A\ \langle \frac{\Sigma}{1/A,B \bullet U} \rangle \quad (S_1\text{:conj:1})$$

$$exit\ A\ \langle \frac{\Sigma}{1/A,B \bullet U} \rangle \rightarrow call\ B\ \langle \frac{\Sigma}{2/A,B \bullet U} \rangle \quad (S_1\text{:conj:2})$$

$$fail\ A\ \langle \frac{\Sigma}{1/A,B \bullet U} \rangle \rightarrow fail\ A, B\ \langle \frac{\Sigma}{U} \rangle \quad (S_1\text{:conj:3})$$

$$exit\ B\ \langle \frac{\Sigma}{2/A,B \bullet U} \rangle \rightarrow exit\ A, B\ \langle \frac{\Sigma}{U} \rangle \quad (S_1\text{:conj:4})$$

$$fail\ B\ \langle \frac{\Sigma}{2/A,B \bullet U} \rangle \rightarrow redo\ A\ \langle \frac{\Sigma}{1/A,B \bullet U} \rangle \quad (S_1\text{:conj:5})$$

(S$_1$:conj:6)

S_1:PP Calculus: **Conjunction**

$$call\ A, B\ \langle \frac{\Sigma}{U} \rangle \rightarrow call\ A\ \langle \frac{\Sigma}{1/A,B \bullet U} \rangle \quad (S_1\text{:conj:1})$$

$$exit\ A\ \langle \frac{\Sigma}{1/A,B \bullet U} \rangle \rightarrow call\ B\ \langle \frac{\Sigma}{2/A,B \bullet U} \rangle \quad (S_1\text{:conj:2})$$

$$fail\ A\ \langle \frac{\Sigma}{1/A,B \bullet U} \rangle \rightarrow fail\ A, B\ \langle \frac{\Sigma}{U} \rangle \quad (S_1\text{:conj:3})$$

$$exit\ B\ \langle \frac{\Sigma}{2/A,B \bullet U} \rangle \rightarrow exit\ A, B\ \langle \frac{\Sigma}{U} \rangle \quad (S_1\text{:conj:4})$$

$$fail\ B\ \langle \frac{\Sigma}{2/A,B \bullet U} \rangle \rightarrow redo\ A\ \langle \frac{\Sigma}{1/A,B \bullet U} \rangle \quad (S_1\text{:conj:5})$$

$$redo\ A, B\ \langle \frac{\Sigma}{U} \rangle \rightarrow redo\ B\ \langle \frac{\Sigma}{2/A,B \bullet U} \rangle \quad (S_1\text{:conj:6})$$

S_1:PP Calculus: **Disjunction**

(S_1:disj:1)

(S_1:disj:2)

(S_1:disj:3)

(S_1:disj:4)

(S_1:disj:5)

S_1:PP Calculus: **Disjunction**

$$call\ A; B\ \langle \frac{\Sigma}{U} \rangle \rightarrowtriangle call\ A\ \langle \frac{\Sigma}{1/A;B \bullet U} \rangle \qquad (S_1\text{:disj:1})$$

(S_1:disj:2)

(S_1:disj:3)

(S_1:disj:4)

(S_1:disj:5)

S_1:PP Calculus: **Disjunction**

$$call\ A; B\ \langle \frac{\Sigma}{U} \rangle \ \rightarrowtriangle\ call\ A\ \langle \frac{\Sigma}{1/A;B \bullet U} \rangle \qquad (S_1\text{:disj:1})$$

$$fail\ A\ \langle \frac{\Sigma}{1/A;B \bullet U} \rangle \ \rightarrowtriangle\ call\ B\ \langle \frac{\Sigma}{2/A;B \bullet U} \rangle \qquad (S_1\text{:disj:2})$$

(S_1:disj:3)

(S_1:disj:4)

(S_1:disj:5)

S_1:PP Calculus: **Disjunction**

$$call\ A; B\ \langle \frac{\Sigma}{U} \rangle \ \rightarrowtriangle\ call\ A\ \langle \frac{\Sigma}{1/A;B \bullet U} \rangle \qquad (S_1\text{:disj:1})$$

$$fail\ A\ \langle \frac{\Sigma}{1/A;B \bullet U} \rangle \ \rightarrowtriangle\ call\ B\ \langle \frac{\Sigma}{2/A;B \bullet U} \rangle \qquad (S_1\text{:disj:2})$$

$$fail\ B\ \langle \frac{\Sigma}{2/A;B \bullet U} \rangle \ \rightarrowtriangle\ fail\ A; B\ \langle \frac{\Sigma}{U} \rangle \qquad (S_1\text{:disj:3})$$

(S_1:disj:4)

(S_1:disj:5)

S_1:PP Calculus: **Disjunction**

$$call\ A; B\ \langle \frac{\boldsymbol{\Sigma}}{\boldsymbol{U}} \rangle \rightarrowtriangle call\ A\ \langle \frac{\boldsymbol{\Sigma}}{1/A;B \bullet \boldsymbol{U}} \rangle \quad (S_1\text{:disj:1})$$

$$fail\ A\ \langle \frac{\boldsymbol{\Sigma}}{1/A;B \bullet \boldsymbol{U}} \rangle \rightarrowtriangle call\ B\ \langle \frac{\boldsymbol{\Sigma}}{2/A;B \bullet \boldsymbol{U}} \rangle \quad (S_1\text{:disj:2})$$

$$fail\ B\ \langle \frac{\boldsymbol{\Sigma}}{2/A;B \bullet \boldsymbol{U}} \rangle \rightarrowtriangle fail\ A; B\ \langle \frac{\boldsymbol{\Sigma}}{\boldsymbol{U}} \rangle \quad (S_1\text{:disj:3})$$

$$exit\ C\ \langle \frac{\boldsymbol{\Sigma}}{N/A;B \bullet \boldsymbol{U}} \rangle \rightarrowtriangle exit\ A; B\ \langle \frac{OR(N) \bullet \boldsymbol{\Sigma}}{\boldsymbol{U}} \rangle, \text{ with } C = \lfloor N/A; B \rfloor \quad (S_1\text{:disj:4})$$

$(S_1\text{:disj:5})$

S_1:PP Calculus: **Disjunction**

$$call\ A; B\ \langle \frac{\Sigma}{U} \rangle \rightarrowtriangle call\ A\ \langle \frac{\Sigma}{1/A;B \bullet U} \rangle \qquad (S_1\text{:disj:1})$$

$$fail\ A\ \langle \frac{\Sigma}{1/A;B \bullet U} \rangle \rightarrowtriangle call\ B\ \langle \frac{\Sigma}{2/A;B \bullet U} \rangle \qquad (S_1\text{:disj:2})$$

$$fail\ B\ \langle \frac{\Sigma}{2/A;B \bullet U} \rangle \rightarrowtriangle fail\ A; B\ \langle \frac{\Sigma}{U} \rangle \qquad (S_1\text{:disj:3})$$

$$exit\ C\ \langle \frac{\Sigma}{N/A;B \bullet U} \rangle \rightarrowtriangle exit\ A; B\ \langle \frac{OR(N) \bullet \Sigma}{U} \rangle, \text{ with } C = \lfloor N/A; B \rfloor \qquad (S_1\text{:disj:4})$$

$$redo\ A; B\ \langle \frac{OR(N) \bullet \Sigma}{U} \rangle \rightarrowtriangle redo\ C\ \langle \frac{\Sigma}{N/A;B \bullet U} \rangle, \text{ with } C = \lfloor N/A; B \rfloor \qquad (S_1\text{:disj:5})$$

Legal event and derivation

- Initial event: $call\ Q\ \langle \frac{nil}{nil} \rangle$, where Q is any goal formula.

Legal event and derivation

- Initial event: $call\ Q\ \langle \frac{nil}{nil} \rangle$, where Q is any goal formula.
- Legal event: reachable from an initial event.

Legal event and derivation

- Initial event: $call\ Q\ \langle \frac{nil}{nil} \rangle$, where Q is any goal formula.
- Legal event: reachable from an initial event.
- Final event: a legal event that does not lead to another event.

$$call\ Q\ \langle \tfrac{nil}{nil} \rangle \quad \rightarrow\!\!\triangleright^* \quad E_0 \ \rightarrow\!\!\triangleright^* \ E$$

Legal event and derivation

- Initial event: $call\ Q\ \langle \frac{nil}{nil} \rangle$, where Q is any goal formula.
- Legal event: reachable from an initial event.
- Final event: a legal event that does not lead to another event.

$$call\ Q\ \langle \tfrac{nil}{nil} \rangle \quad \rightarrow\!\!\triangleright^* \quad \underbrace{E_0 \rightarrow\!\!\triangleright^* E}_{\text{legal derivation}}$$

Legal event and derivation

- Initial event: $call\ Q\ \langle \frac{nil}{nil} \rangle$, where Q is any goal formula.
- Legal event: reachable from an initial event.
- Final event: a legal event that does not lead to another event.

$$call\ Q\ \langle \tfrac{nil}{nil} \rangle \quad \rightarrow^* \quad \underbrace{E_0 \rightarrow^* E}_{\text{legal derivation}} \qquad E: \text{legal event}$$

In S_1:PP, a legal event can have only one legal predecessor, and only one successor (Theorem 4.2 in [Kul04]).

$$
\begin{aligned}
&\mathit{call}\ \mathsf{main}\ ,\ \{\mathit{nil}\},\ \{\mathit{nil}\}\\
&\quad\twoheadrightarrow\ \mathit{call}\ (\mathsf{good},\mathsf{bad})\ ,\ \{\mathsf{main}\bullet\mathit{nil}\},\ \{\mathit{nil}\}\\
&\quad\twoheadrightarrow\ \mathit{call}\ \mathsf{good}\ ,\ \{\mathit{1}/\mathsf{good},\mathsf{bad}\bullet\mathsf{main}\bullet\mathit{nil}\},\ \{\mathit{nil}\}\\
&\quad\twoheadrightarrow\ \mathit{call}\ \mathsf{true}\ ,\ \{\mathsf{good}\bullet\mathit{1}/\mathsf{good},\mathsf{bad}\bullet\mathsf{main}\bullet\mathit{nil}\},\ \{\mathit{nil}\}\\
&\quad\twoheadrightarrow\ \mathit{exit}\ \mathsf{true}\ ,\ \{\mathsf{good}\bullet\mathit{1}/\mathsf{good},\mathsf{bad}\bullet\mathsf{main}\bullet\mathit{nil}\},\ \{\mathit{nil}\}\\
&\quad\twoheadrightarrow\ \mathit{exit}\ \mathsf{good}\ ,\ \{\mathit{1}/\mathsf{good},\mathsf{bad}\bullet\mathsf{main}\bullet\mathit{nil}\},\ \{\mathit{BY}(\mathsf{true})\bullet\mathit{nil}\}\\
&\quad\twoheadrightarrow\ \mathit{call}\ \mathsf{bad}\ ,\ \{\mathit{2}/\mathsf{good},\mathsf{bad}\bullet\mathsf{main}\bullet\mathit{nil}\},\ \{\mathit{BY}(\mathsf{true})\bullet\mathit{nil}\}\\
&\quad\twoheadrightarrow\ \mathit{fail}\ \mathsf{bad}\ ,\ \{\mathit{2}/\mathsf{good},\mathsf{bad}\bullet\mathsf{main}\bullet\mathit{nil}\},\ \{\mathit{BY}(\mathsf{true})\bullet\mathit{nil}\}\\
&\quad\twoheadrightarrow\ \mathit{redo}\ \mathsf{good}\ ,\ \{\mathit{1}/\mathsf{good},\mathsf{bad}\bullet\mathsf{main}\bullet\mathit{nil}\},\ \{\mathit{BY}(\mathsf{true})\bullet\mathit{nil}\}\\
&\quad\twoheadrightarrow\ \mathit{redo}\ \mathsf{true}\ ,\ \{\mathsf{good}\bullet\mathit{1}/\mathsf{good},\mathsf{bad}\bullet\mathsf{main}\bullet\mathit{nil}\},\ \{\mathit{nil}\}\\
&\quad\twoheadrightarrow\ \mathit{fail}\ \mathsf{true}\ ,\ \{\mathsf{good}\bullet\mathit{1}/\mathsf{good},\mathsf{bad}\bullet\mathsf{main}\bullet\mathit{nil}\},\ \{\mathit{nil}\}\\
&\quad\twoheadrightarrow\ \mathit{fail}\ \mathsf{good}\ ,\ \{\mathit{1}/\mathsf{good},\mathsf{bad}\bullet\mathsf{main}\bullet\mathit{nil}\},\ \{\mathit{nil}\}\\
&\quad\twoheadrightarrow\ \mathit{fail}\ (\mathsf{good},\mathsf{bad})\ ,\ \{\mathsf{main}\bullet\mathit{nil}\},\ \{\mathit{nil}\}\\
&\quad\twoheadrightarrow\ \mathit{fail}\ \mathsf{main}\ ,\ \{\mathit{nil}\},\ \{\mathit{nil}\}
\end{aligned}
$$

$$
\begin{aligned}
&\mathit{call}\ \mathsf{main}\ ,\ \{\mathit{nil}\},\ \{\mathit{nil}\}\\
&\rightarrowtriangle\ \mathit{call}\ (\mathsf{good},\mathsf{bad})\ ,\ \{\mathsf{main}\bullet\mathit{nil}\},\ \{\mathit{nil}\}\\
&\rightarrowtriangle\ \mathit{call}\ \mathsf{good}\ ,\ \{\mathit{1}/\mathsf{good},\mathsf{bad}\bullet\mathsf{main}\bullet\mathit{nil}\},\ \{\mathit{nil}\}\\
&\rightarrowtriangle\ \mathit{call}\ \mathsf{true}\ ,\ \{\mathsf{good}\bullet\mathit{1}/\mathsf{good},\mathsf{bad}\bullet\mathsf{main}\bullet\mathit{nil}\},\ \{\mathit{nil}\}\\
&\rightarrowtriangle\ \mathit{exit}\ \mathsf{true}\ ,\ \{\mathsf{good}\bullet\mathit{1}/\mathsf{good},\mathsf{bad}\bullet\mathsf{main}\bullet\mathit{nil}\},\ \{\mathit{nil}\}\\
&\rightarrowtriangle\ \mathit{exit}\ \mathsf{good}\ ,\ \{\mathit{1}/\mathsf{good},\mathsf{bad}\bullet\mathsf{main}\bullet\mathit{nil}\},\ \{\mathit{BY}(\mathsf{true})\bullet\mathit{nil}\}\\
&\rightarrowtriangle\ \mathit{call}\ \mathsf{bad}\ ,\ \{\mathit{2}/\mathsf{good},\mathsf{bad}\bullet\mathsf{main}\bullet\mathit{nil}\},\ \{\mathit{BY}(\mathsf{true})\bullet\mathit{nil}\}\\
&\rightarrowtriangle\ \mathit{fail}\ \mathsf{bad}\ ,\ \{\mathit{2}/\mathsf{good},\mathsf{bad}\bullet\mathsf{main}\bullet\mathit{nil}\},\ \{\mathit{BY}(\mathsf{true})\bullet\mathit{nil}\}\\
&\rightarrowtriangle\ \mathit{redo}\ \mathsf{good}\ ,\ \{\mathit{1}/\mathsf{good},\mathsf{bad}\bullet\mathsf{main}\bullet\mathit{nil}\},\ \{\mathit{BY}(\mathsf{true})\bullet\mathit{nil}\}\\
&\rightarrowtriangle\ \mathit{redo}\ \mathsf{true}\ ,\ \{\mathsf{good}\bullet\mathit{1}/\mathsf{good},\mathsf{bad}\bullet\mathsf{main}\bullet\mathit{nil}\},\ \{\mathit{nil}\}\\
&\rightarrowtriangle\ \mathit{fail}\ \mathsf{true}\ ,\ \{\mathsf{good}\bullet\mathit{1}/\mathsf{good},\mathsf{bad}\bullet\mathsf{main}\bullet\mathit{nil}\},\ \{\mathit{nil}\}\\
&\rightarrowtriangle\ \mathit{fail}\ \mathsf{good}\ ,\ \{\mathit{1}/\mathsf{good},\mathsf{bad}\bullet\mathsf{main}\bullet\mathit{nil}\},\ \{\mathit{nil}\}\\
&\rightarrowtriangle\ \mathit{fail}\ (\mathsf{good},\mathsf{bad})\ ,\ \{\mathsf{main}\bullet\mathit{nil}\},\ \{\mathit{nil}\}\\
&\rightarrowtriangle\ \mathit{fail}\ \mathsf{main}\ ,\ \{\mathit{nil}\},\ \{\mathit{nil}\}
\end{aligned}
$$

$call$ main , $\{nil\}$, $\{nil\}$

$\rightarrowtriangle$ $call$ (good, bad) , $\{\text{main} \bullet nil\}$, $\{nil\}$

$\rightarrowtriangle$ $call$ good , $\{1/\text{good}, \text{bad} \bullet \text{main} \bullet nil\}$, $\{nil\}$

$\rightarrowtriangle$ $call$ true , $\{\text{good} \bullet 1/\text{good}, \text{bad} \bullet \text{main} \bullet nil\}$, $\{nil\}$

$\rightarrowtriangle$ $exit$ true , $\{\text{good} \bullet 1/\text{good}, \text{bad} \bullet \text{main} \bullet nil\}$, $\{nil\}$

$\rightarrowtriangle$ $exit$ good , $\{1/\text{good}, \text{bad} \bullet \text{main} \bullet nil\}$, $\{BY(\text{true}) \bullet nil\}$

$\rightarrowtriangle$ $call$ bad , $\{2/\text{good}, \text{bad} \bullet \text{main} \bullet nil\}$, $\{BY(\text{true}) \bullet nil\}$

$\rightarrowtriangle$ $fail$ bad , $\{2/\text{good}, \text{bad} \bullet \text{main} \bullet nil\}$, $\{BY(\text{true}) \bullet nil\}$

$\rightarrowtriangle$ $redo$ good , $\{1/\text{good}, \text{bad} \bullet \text{main} \bullet nil\}$, $\{BY(\text{true}) \bullet nil\}$

$\rightarrowtriangle$ $redo$ true , $\{\text{good} \bullet 1/\text{good}, \text{bad} \bullet \text{main} \bullet nil\}$, $\{nil\}$

$\rightarrowtriangle$ $fail$ true , $\{\text{good} \bullet 1/\text{good}, \text{bad} \bullet \text{main} \bullet nil\}$, $\{nil\}$

$\rightarrowtriangle$ $fail$ good , $\{1/\text{good}, \text{bad} \bullet \text{main} \bullet nil\}$, $\{nil\}$

$\rightarrowtriangle$ $fail$ (good, bad) , $\{\text{main} \bullet nil\}$, $\{nil\}$

$\rightarrowtriangle$ $fail$ main , $\{nil\}$, $\{nil\}$

Aggregations: Simple pass

- forward derivation relative to U: $Push\ G\ \langle \frac{\Sigma}{U} \rangle \underbrace{\rightarrow \rightarrow \ldots \rightarrow}_{\text{no } Pop\_\langle \frac{\bar{\Sigma}}{U} \rangle} E$

Aggregations: Simple pass

- forward derivation relative to $\boldsymbol{U}$: $Push\ G\ \langle \frac{\boldsymbol{\Sigma}}{\boldsymbol{U}} \rangle \overset{\triangleright}{\longrightarrow} E$

Aggregations: Simple pass

- forward derivation relative to $\boldsymbol{U}$: $Push\ G\ \langle \frac{\boldsymbol{\Sigma}}{\boldsymbol{U}} \rangle \overset{\triangleright}{\longrightarrow} E$

- backward derivation relative to $\boldsymbol{U}$: $Pop\ G\ \langle \frac{\boldsymbol{\Sigma}}{\boldsymbol{U}} \rangle \underbrace{\longrightarrow \longrightarrow \ldots \longrightarrow}_{\text{no } Push\ \_\ \langle \frac{=}{\boldsymbol{U}} \rangle} E$

Aggregations: Simple pass

- forward derivation relative to $\boldsymbol{U}$: $Push\ G\ \langle \frac{\boldsymbol{\Sigma}}{\boldsymbol{U}} \rangle \xrightarrow{\triangleright} E$
- backward derivation relative to $\boldsymbol{U}$: $Pop\ G\ \langle \frac{\boldsymbol{\Sigma}}{\boldsymbol{U}} \rangle \xrightarrow{\triangleleft} E$

Aggregations: Simple pass

- forward derivation relative to $\boldsymbol{U}$: $Push\ G\ \langle \frac{\boldsymbol{\Sigma}}{\boldsymbol{U}} \rangle \xrightarrow{\triangleright} E$
- backward derivation relative to $\boldsymbol{U}$: $Pop\ G\ \langle \frac{\boldsymbol{\Sigma}}{\boldsymbol{U}} \rangle \xrightarrow{\triangleleft} E$
- simple pass relative to G,$\boldsymbol{U}$: $Push\ G\ \langle \frac{=}{\boldsymbol{U}} \rangle \xrightarrow{\triangleright} Pop\ G\ \langle \frac{=}{\boldsymbol{U}} \rangle$
  $Pop\ G\ \langle \frac{=}{\boldsymbol{U}} \rangle \xrightarrow{\triangleleft} Push\ G\ \langle \frac{=}{\boldsymbol{U}} \rangle$

Aggregations: Simple pass

- forward derivation relative to $\boldsymbol{U}$: $Push\ G\ \langle \frac{\boldsymbol{\Sigma}}{\boldsymbol{U}} \rangle \overset{\triangleright}{\longrightarrow} E$

- backward derivation relative to $\boldsymbol{U}$: $Pop\ G\ \langle \frac{\boldsymbol{\Sigma}}{\boldsymbol{U}} \rangle \overset{\triangleleft}{\longrightarrow} E$

- simple pass relative to G,$\boldsymbol{U}$:

$$\begin{array}{lcl} Push\ G\ \langle \frac{=}{\boldsymbol{U}} \rangle & \longrightarrow & Pop\ G\ \langle \frac{=}{\boldsymbol{U}} \rangle \\ Pop\ G\ \langle \frac{=}{\boldsymbol{U}} \rangle & \longrightarrow & Push\ G\ \langle \frac{=}{\boldsymbol{U}} \rangle \\ \_\ G\ \langle \frac{=}{\boldsymbol{U}} \rangle & \longrightarrow & \_\ G\ \langle \frac{=}{\boldsymbol{U}} \rangle \\ \text{stage} & & \text{stage} \end{array}$$

Aggregations: Composed pass

Based on:

- If $Pop\ G\ \langle \frac{\Sigma}{U} \rangle$ is legal, then $Pop\ G\ \langle \frac{\Sigma}{U} \rangle \longleftarrow Push\ G\ \langle \frac{\Sigma^{\circ}}{U} \rangle$. (Lemma 5.6)
- If $fail\ H\ \langle \frac{\Sigma}{U} \rangle$ is legal, then $fail\ H\ \langle \frac{\Sigma}{U} \rangle \Leftarrow^{*} call\ H\ \langle \frac{\Sigma}{U} \rangle$, where $call\ H\ \langle \frac{=}{U} \rangle$ does not appear within the derivation. Furthermore, if $redo\ H\ \langle \frac{\Sigma}{U} \rangle$ is legal, then $redo\ H\ \langle \frac{\Sigma}{U} \rangle \longleftarrow exit\ H\ \langle \frac{\Sigma}{U} \rangle$. (Theorem 6.1)

Aggregations: Composed pass

Based on:

- If $Pop\ G\ \langle \frac{\Sigma}{U} \rangle$ is legal, then $Pop\ G\ \langle \frac{\Sigma}{U} \rangle \longleftarrow Push\ G\ \langle \frac{\Sigma^{\circ}}{U} \rangle$. (Lemma 5.6)
- If $fail\ H\ \langle \frac{\Sigma}{U} \rangle$ is legal, then $fail\ H\ \langle \frac{\Sigma}{U} \rangle \Leftarrow^{*} call\ H\ \langle \frac{\Sigma}{U} \rangle$, where $call\ H\ \langle \frac{=}{U} \rangle$ does not appear within the derivation. Furthermore, if $redo\ H\ \langle \frac{\Sigma}{U} \rangle$ is legal, then $redo\ H\ \langle \frac{\Sigma}{U} \rangle \longleftarrow exit\ H\ \langle \frac{\Sigma}{U} \rangle$. (Theorem 6.1)

$redo\ G\ \langle \frac{=}{U} \rangle$

Aggregations: Composed pass

Based on:

- If $Pop\ G\ \langle \frac{\Sigma}{U} \rangle$ is legal, then $Pop\ G\ \langle \frac{\Sigma}{U} \rangle \longleftarrow Push\ G\ \langle \frac{\Sigma^{\circ}}{U} \rangle$. (Lemma 5.6)
- If $fail\ H\ \langle \frac{\Sigma}{U} \rangle$ is legal, then $fail\ H\ \langle \frac{\Sigma}{U} \rangle \Leftarrow^{*} call\ H\ \langle \frac{\Sigma}{U} \rangle$, where $call\ H\ \langle \frac{=}{U} \rangle$ does not appear within the derivation. Furthermore, if $redo\ H\ \langle \frac{\Sigma}{U} \rangle$ is legal, then $redo\ H\ \langle \frac{\Sigma}{U} \rangle \longleftarrow exit\ H\ \langle \frac{\Sigma}{U} \rangle$. (Theorem 6.1)

$redo\ G\ \langle \frac{=}{U} \rangle \longleftarrow exit\ G\ \langle \frac{=}{U} \rangle$

Aggregations: Composed pass

Based on:

- If $Pop\ G\ \langle \frac{\Sigma}{U} \rangle$ is legal, then $Pop\ G\ \langle \frac{\Sigma}{U} \rangle \longleftarrow Push\ G\ \langle \frac{\Sigma^{\circ}}{U} \rangle$. (Lemma 5.6)
- If $fail\ H\ \langle \frac{\Sigma}{U} \rangle$ is legal, then $fail\ H\ \langle \frac{\Sigma}{U} \rangle \Leftarrow^{*} call\ H\ \langle \frac{\Sigma}{U} \rangle$, where $call\ H\ \langle \frac{=}{U} \rangle$ does not appear within the derivation. Furthermore, if $redo\ H\ \langle \frac{\Sigma}{U} \rangle$ is legal, then $redo\ H\ \langle \frac{\Sigma}{U} \rangle \longleftarrow exit\ H\ \langle \frac{\Sigma}{U} \rangle$. (Theorem 6.1)

$redo\ G\ \langle \frac{=}{U} \rangle \longleftarrow exit\ G\ \langle \frac{=}{U} \rangle \longleftarrow Push\ G\ \langle \frac{=}{U} \rangle$

Aggregations: Composed pass

Based on:

- If $Pop\ G\ \langle \frac{\Sigma}{U} \rangle$ is legal, then $Pop\ G\ \langle \frac{\Sigma}{U} \rangle \longleftarrow Push\ G\ \langle \frac{\Sigma^{\circ}}{U} \rangle$. (Lemma 5.6)
- If $fail\ H\ \langle \frac{\Sigma}{U} \rangle$ is legal, then $fail\ H\ \langle \frac{\Sigma}{U} \rangle \Leftarrow^{*} call\ H\ \langle \frac{\Sigma}{U} \rangle$, where $call\ H\ \langle \frac{=}{U} \rangle$ does not appear within the derivation. Furthermore, if $redo\ H\ \langle \frac{\Sigma}{U} \rangle$ is legal, then $redo\ H\ \langle \frac{\Sigma}{U} \rangle \longleftarrow exit\ H\ \langle \frac{\Sigma}{U} \rangle$. (Theorem 6.1)

$$redo\ G\ \langle \tfrac{=}{U} \rangle \longleftarrow exit\ G\ \langle \tfrac{=}{U} \rangle \longleftarrow Push\ G\ \langle \tfrac{=}{U} \rangle \longleftarrow \ldots \longleftarrow call\ G\ \langle \tfrac{=}{U} \rangle$$

Aggregations: Composed pass

Based on:

- If $Pop\ G\ \langle \frac{\boldsymbol{\Sigma}}{\boldsymbol{U}} \rangle$ is legal, then $Pop\ G\ \langle \frac{\boldsymbol{\Sigma}}{\boldsymbol{U}} \rangle \longleftarrow Push\ G\ \langle \frac{\boldsymbol{\Sigma}^\circ}{\boldsymbol{U}} \rangle$. (Lemma 5.6)
- If $fail\ H\ \langle \frac{\boldsymbol{\Sigma}}{\boldsymbol{U}} \rangle$ is legal, then $fail\ H\ \langle \frac{\boldsymbol{\Sigma}}{\boldsymbol{U}} \rangle \leftarrow^* call\ H\ \langle \frac{\boldsymbol{\Sigma}}{\boldsymbol{U}} \rangle$, where $call\ H\ \langle \frac{=}{\boldsymbol{U}} \rangle$ does not appear within the derivation. Furthermore, if $redo\ H\ \langle \frac{\boldsymbol{\Sigma}}{\boldsymbol{U}} \rangle$ is legal, then $redo\ H\ \langle \frac{\boldsymbol{\Sigma}}{\boldsymbol{U}} \rangle \longleftarrow exit\ H\ \langle \frac{\boldsymbol{\Sigma}}{\boldsymbol{U}} \rangle$. (Theorem 6.1)

$$redo\ G\ \langle \tfrac{=}{\boldsymbol{U}} \rangle \longleftarrow exit\ G\ \langle \tfrac{=}{\boldsymbol{U}} \rangle \longleftarrow Push\ G\ \langle \tfrac{=}{\boldsymbol{U}} \rangle \longleftarrow \ldots \longleftarrow call\ G\ \langle \tfrac{=}{\boldsymbol{U}} \rangle$$

Definition of composed pass relative to $G,\boldsymbol{U}$:

$$E_1^{G,\boldsymbol{U}} \quad \underbrace{\longrightarrow\longrightarrow \ldots \longrightarrow}_{\text{no } call\ G\ \langle \frac{=}{\boldsymbol{U}} \rangle} \quad E_2^{G,\boldsymbol{U}}$$

also called composable sequence

Aggregations: Composed pass

Based on:

- If $Pop\ G\ \langle \frac{\Sigma}{U} \rangle$ is legal, then $Pop\ G\ \langle \frac{\Sigma}{U} \rangle \longleftarrow Push\ G\ \langle \frac{\Sigma^{\circ}}{U} \rangle$. (Lemma 5.6)
- If $fail\ H\ \langle \frac{\Sigma}{U} \rangle$ is legal, then $fail\ H\ \langle \frac{\Sigma}{U} \rangle \Leftarrow^{*} call\ H\ \langle \frac{\Sigma}{U} \rangle$, where $call\ H\ \langle \frac{=}{U} \rangle$ does not appear within the derivation. Furthermore, if $redo\ H\ \langle \frac{\Sigma}{U} \rangle$ is legal, then $redo\ H\ \langle \frac{\Sigma}{U} \rangle \longleftarrow exit\ H\ \langle \frac{\Sigma}{U} \rangle$. (Theorem 6.1)

$$redo\ G\ \langle \tfrac{=}{U} \rangle \longleftarrow exit\ G\ \langle \tfrac{=}{U} \rangle \longleftarrow Push\ G\ \langle \tfrac{=}{U} \rangle \longleftarrow \ldots \longleftarrow call\ G\ \langle \tfrac{=}{U} \rangle$$

Definition of composed pass relative to G,U:

$$E_1^{G,U} \Longrightarrow E_2^{G,U}$$

Some properties of S_1:PP

Uniqueness of steps: transition relation, as well as its converse on legal events, are functional (Theorem 4.2)

$\longrightarrow$ forward and backward derivation steps are possible

For application, see the sketches on non-interference and modularity of conjunction.

Independence on the context of derivation: the same goal goes through the same stages (Theorem 7.6)

Theorem 7.6 (independence) *Let the following sequences be composable, where* $m, k \geq 1$, *and let* $\boldsymbol{\Theta}(H) = \boldsymbol{\Sigma}(G)$.

$$call\ G\ \langle \tfrac{\boldsymbol{\Sigma}}{\boldsymbol{U}} \rangle \longrightarrow E_1^{G,\boldsymbol{U}} \longrightarrow \ldots \longrightarrow E_m^{G,\boldsymbol{U}}$$
$$call\ H\ \langle \tfrac{\boldsymbol{\Theta}}{\boldsymbol{V}} \rangle \longrightarrow E_1^{H,\boldsymbol{V}} \longrightarrow \ldots \longrightarrow E_k^{H,\boldsymbol{V}}$$

Then for any i *in common* ($i \leq m, k$) *holds:*

$$\textit{If } E_i^{G,\boldsymbol{U}} := \Gamma\ G\ \langle \tfrac{\boldsymbol{\Sigma}'}{\boldsymbol{U}} \rangle \ \textit{ then } \ E_i^{H,\boldsymbol{V}} = \Gamma\ H\ \langle \tfrac{\boldsymbol{\Sigma}' - \boldsymbol{\Sigma} + \boldsymbol{\Theta}}{\boldsymbol{V}} \rangle.$$

A legal derivation is modular, due to compositionality of rules wrt conjunction and disjunction, and stages (Theorem 7.2, Theorem 7.3)

$\longrightarrow$ abstracting parts of a derivation is possible

Theorem 7.2 ($\oplus$) *If* $call\ G\ \langle \frac{nil}{nil} \rangle \rightarrowtriangle E_1 \rightarrowtriangle \ldots \rightarrowtriangle E_n \rightarrowtriangle Pop\ G\ \langle \frac{\boldsymbol{\Delta}}{nil} \rangle$, *then for every* G°, $\boldsymbol{U}$, $\boldsymbol{\Sigma}$ *such that* $call\ G^\circ\ \langle \frac{\boldsymbol{\Sigma}}{\boldsymbol{U}} \rangle$ *is legal and* $\boldsymbol{\Sigma}(G^\circ) = G$ *holds:*

$$call\ G^\circ\ \langle \tfrac{\boldsymbol{\Sigma}}{\boldsymbol{U}} \rangle \rightarrowtriangle E_1{}^{\S} \oplus \langle \tfrac{\boldsymbol{\Sigma}}{\boldsymbol{U}} \rangle \rightarrowtriangle \ldots \rightarrowtriangle E_n{}^{\S} \oplus \langle \tfrac{\boldsymbol{\Sigma}}{\boldsymbol{U}} \rangle \rightarrowtriangle Pop\ G^\circ\ \langle \tfrac{\boldsymbol{\Delta} + \boldsymbol{\Sigma}}{\boldsymbol{U}} \rangle$$

is also a legal derivation.

Theorem 7.3 ($\ominus$) *Let* $call\ G\ \langle \frac{\boldsymbol{\Sigma}}{\boldsymbol{U}} \rangle \rightarrowtriangle E_1 \rightarrowtriangle \ldots \rightarrowtriangle E_n \rightarrowtriangle Pop\ G\ \langle \frac{\boldsymbol{\Omega}}{\boldsymbol{U}} \rangle$ *be legal, and* $E_i \neq Pop\ \_\ \langle \frac{=}{\boldsymbol{U}} \rangle$ *for every* i. *Then for* $G' := \boldsymbol{\Sigma}(G)$ *holds:*

$$call\ G'\ \langle \tfrac{nil}{nil} \rangle \rightarrowtriangle E'_1 \ominus \langle \tfrac{\boldsymbol{\Sigma}}{\boldsymbol{U}} \rangle \rightarrowtriangle \ldots \rightarrowtriangle E'_n \ominus \langle \tfrac{\boldsymbol{\Sigma}}{\boldsymbol{U}} \rangle \rightarrowtriangle Pop\ G'\ \langle \tfrac{\boldsymbol{\Omega} - \boldsymbol{\Sigma}}{nil} \rangle$$

is also a legal derivation. Here if $E_i = \Gamma\ H\ \langle \frac{\boldsymbol{\Theta}}{\boldsymbol{V}} \rangle$, *then* $E'_i := \Gamma\ H'\ \langle \frac{\boldsymbol{\Theta}}{\boldsymbol{V}'} \rangle$, *where* $\boldsymbol{\Sigma}(H) = H'$ *and* $\boldsymbol{\Sigma}(\boldsymbol{V}) = \boldsymbol{V}'$.

---

§ possibly the fresh variables of E_i have to be renamed

Modeling Prolog execution in S_1:PP

Theorem 8.1 (existential termination, success, failure) *Prolog computation of a goal G relative to Π terminates existentially if there is a simple pass*

$$call\ G\ \langle \tfrac{nil}{nil} \rangle \longrightarrow Pop\ G\ \langle \tfrac{\mathbf{\Delta}}{nil} \rangle$$

In case $Pop = exit$, the computation is successful, otherwise it is failed.

Theorem 8.2 (the first computed answer) *If $call\ G\ \langle \tfrac{nil}{nil} \rangle \longrightarrow exit\ G\ \langle \tfrac{\mathbf{\Delta}}{nil} \rangle$, then $subst(\mathbf{\Delta})\mid_G$ is the first computed answer substitution for G relative to Π in Prolog.*

Theorem 8.3 (all computed answers, universal termination) *In a composable sequence*

$$call\ G\ \langle \tfrac{nil}{1/G,\mathsf{fail} \bullet nil} \rangle \longrightarrow^{2k-1} exit\ G\ \langle \tfrac{\mathbf{\Delta}}{1/G,\mathsf{fail} \bullet nil} \rangle$$

is $subst(\mathbf{\Delta})\mid_G$ the kth computed answer substitution for G relative to Π in Prolog. Furthermore, G is universally terminating relative to Π if

$$call\ G\ \langle \tfrac{nil}{1/G,\mathsf{fail} \bullet nil} \rangle \Longrightarrow fail\ G\ \langle \tfrac{nil}{1/G,\mathsf{fail} \bullet nil} \rangle$$

Vanilla meta-interpreter

Let $exit\ A, B\ \langle \frac{\boldsymbol{\Sigma}}{\boldsymbol{U}} \rangle$ be a legal event. How could it have been derived? There is only one way:

$exit\ A, B\ \langle \frac{\boldsymbol{\Sigma}}{\boldsymbol{U}} \rangle$

$\Leftarrow$ $exit\ B\ \langle \frac{\boldsymbol{\Sigma}}{2/A,B \bullet \boldsymbol{U}} \rangle$, by ($S_1$:conj:4)

$\Longleftarrow$ $call\ B\ \langle \frac{\boldsymbol{\Sigma}^\circ}{2/A,B \bullet \boldsymbol{U}} \rangle$, by Theorem 6.5(2), with $\boldsymbol{\Sigma} \succeq \boldsymbol{\Sigma}^\circ$

$\Leftarrow$ $exit\ A\ \langle \frac{\boldsymbol{\Sigma}^\circ}{1/A,B \bullet \boldsymbol{U}} \rangle$, by ($S_1$:conj:2)

$\Longleftarrow$ $call\ A\ \langle \frac{\boldsymbol{\Sigma}^{\circ\circ}}{1/A,B \bullet \boldsymbol{U}} \rangle$, by Theorem 6.5(2), with $\boldsymbol{\Sigma}^\circ \succeq \boldsymbol{\Sigma}^{\circ\circ}$

$\Leftarrow$ $call\ A, B\ \langle \frac{\boldsymbol{\Sigma}^{\circ\circ}}{\boldsymbol{U}} \rangle$, by ($S_1$:conj:1)

The converse (starting from $exit\ A$ and $exit\ B$) yields itself as well. Similarly for disjunction.

Summary of S_1:PP operational semantics

- focus on general goals[¶] instead of selected atoms
- forward and backward derivation steps treated equally
- modularity of derivation: parts of execution can be abstracted
- lean data structure for "state of computation":
  - one ancestor stack, A-stack
  - one (generalized) environment, B-stack

[¶] there are precursors in advocating the idea of general goals: [CvEHL92], [TB93]

Back to motivation: Capturing non-interference

Push events (call, redo) are more amenable to forward steps, and pop events (exit, fail) are more amenable to backward steps.

Example: Assume run-time test Cc succeeds exactly once.

$$
\begin{array}{l}
call\ Cc, G\ \langle \frac{=}{U} \rangle \\
\quad \rightarrowtriangle \quad call\ Cc\ \langle \frac{=}{Cc, G \bullet U} \rangle \\
\quad \longrightarrow \quad exit\ Cc\ \langle \frac{=}{Cc, G \bullet U} \rangle \\
\quad \rightarrowtriangle \quad call\ G\ \langle \frac{=}{Cc, G \bullet U} \rangle
\end{array}
\qquad
\begin{array}{l}
fail\ Cc, G\ \langle \frac{=}{U} \rangle \\
\quad \lefttriangle \quad fail\ Cc\ \langle \frac{=}{Cc, G \bullet U} \rangle \\
\quad \longleftarrow \quad redo\ Cc\ \langle \frac{=}{Cc, G \bullet U} \rangle \\
\quad \lefttriangle \quad fail\ G\ \langle \frac{=}{Cc, G \bullet U} \rangle
\end{array}
$$

$$
\begin{array}{l}
exit\ Cc, G\ \langle \frac{=}{U} \rangle \\
\quad \lefttriangle \quad exit\ G\ \langle \frac{=}{Cc, G \bullet U} \rangle
\end{array}
\qquad
\begin{array}{l}
redo\ Cc, G\ \langle \frac{=}{U} \rangle \\
\quad \rightarrowtriangle \quad redo\ G\ \langle \frac{=}{Cc, G \bullet U} \rangle
\end{array}
$$

References

[Byr80] Lawrence Byrd. Understanding the control flow of Prolog programs. In S. A. Tärnlund, editor, Proc. of the 1980 Logic Programming Workshop, Debrecen, pages 127–138, 1980. Also as D.A.I. Research Paper 151.

[CvEHL92] M. H. M. Cheng, M. H. van Emden, R. N. Horspool, and M. Levy. Compositional operational semantics for Prolog programs. J. New Generation Computing, 10(3):315–328, 1992.

[JM84] N. D. Jones and A. Mycroft. Stepwise development of operational and denotational semantics for Prolog. In Proc. of ISLP'84, Atlantic City, pages 281–288, 1984.

[Kul04] M. Kulaš. Toward the concept of backtracking computation. In L. Aceto, W. J. Fokkink, and I. Ulidowski, editors, Proc. of SOS'04, London, ENTCS. Elsevier, 2004. To appear.

[TB93] G. Tobermann and C. Beckstein. What's in a trace: The box model revisited. In Proc. of AADEBUG'93, Linköping, volume 749 of LNCS, pages 171–187. Springer-Verlag, 1993.