

Make FunBlocks alive

Marvin FOURASTIE

Master project

Motivations



Educational purpose



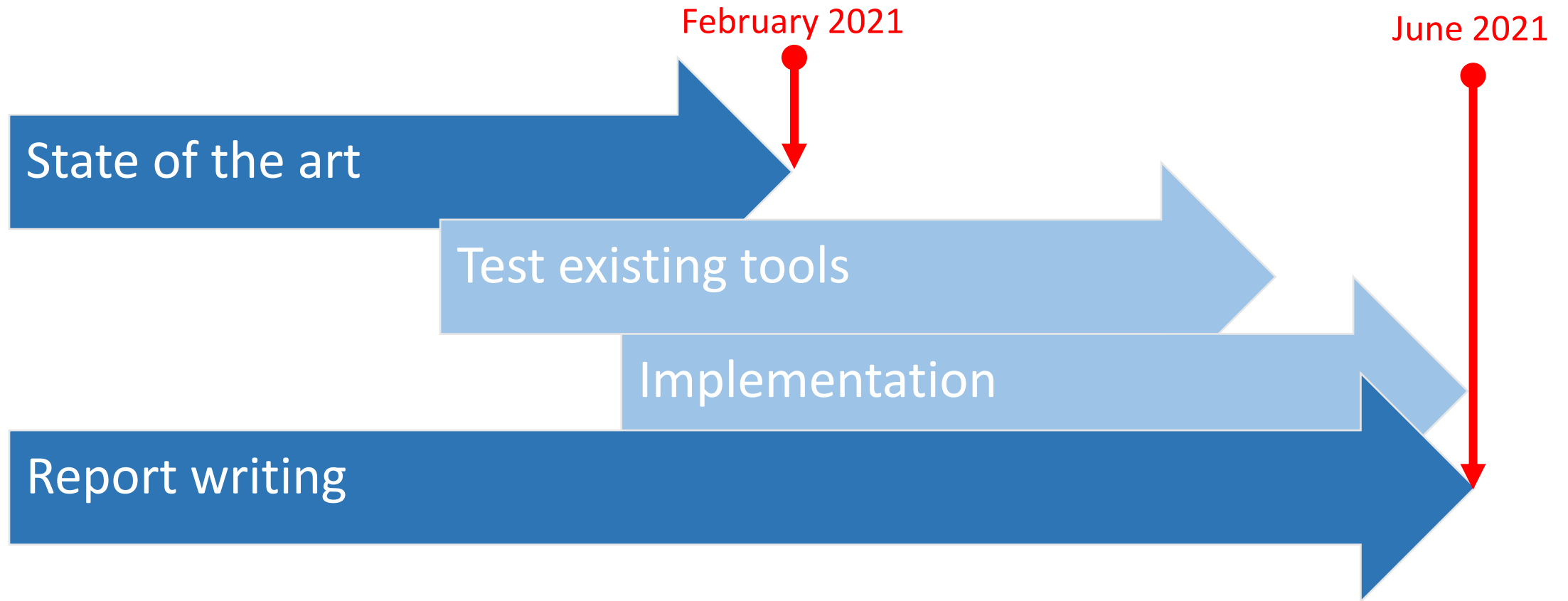
Imperative paradigm alternative



Based on rewrite systems

FunBlocks

Road Map



FunBlocks

```
1  init area(disk(r))  
2  
3  case area(disk($d)) => mul(pi,square($d))
```

PROGRAM STATE

area disk r

RULES

▶ area disk < d > → mul pi square < d >

PROGRAM STATE

mul pi square r

RULES

▶ area disk < d > → mul pi square < d >

FunBlocks

Declarative paradigm



```
area(disk(r))
```

Based on rewrite systems



```
case area(disk($d)) => mul(pi,square($d))
```

Static typing



```
type Tree $t :: empty | leaf $t | node (Tree $t) (Tree $t)
```

Goals

Provide users with **valuable insights** about their program

→ Verification of rewrite systems

Rewrite systems

Stack operators

$\text{Zero} = \{0\}$

$\text{Nat} = \text{Zero} \cup \text{succ}(\text{Nat})$

$\text{Empty} = \Lambda$

$\text{Stack} = \text{Empty} \cup \text{push}(\text{Nat}, \text{Stack})$

$\text{top} : \text{Stack} \rightarrow \text{Nat}$

$\text{pop} : \text{Stack} \rightarrow \text{Stack}$

$\text{alternate} : \text{Stack} \times \text{Stack} \rightarrow \text{Stack}$

Rewrite systems

Canonical rewrite system

$\text{top}(\text{push}(x, y)) = x$

$\text{pop}(\text{push}(x, y)) = y$

$\text{alternate}(\Lambda, z) = z$

$\text{alternate}(\text{push}(x, y), z) = \text{push}(x, \text{alternate}(z, y))$



$\text{top}(\text{push}(x, y)) \rightarrow x$

$\text{pop}(\text{push}(x, y)) \rightarrow y$

$\text{alternate}(\Lambda, z) \rightarrow z$

$\text{alternate}(\text{push}(x, y), z) \rightarrow \text{push}(x, \text{alternate}(z, y))$

Rewrite systems

Termination

Confluence

Soundness

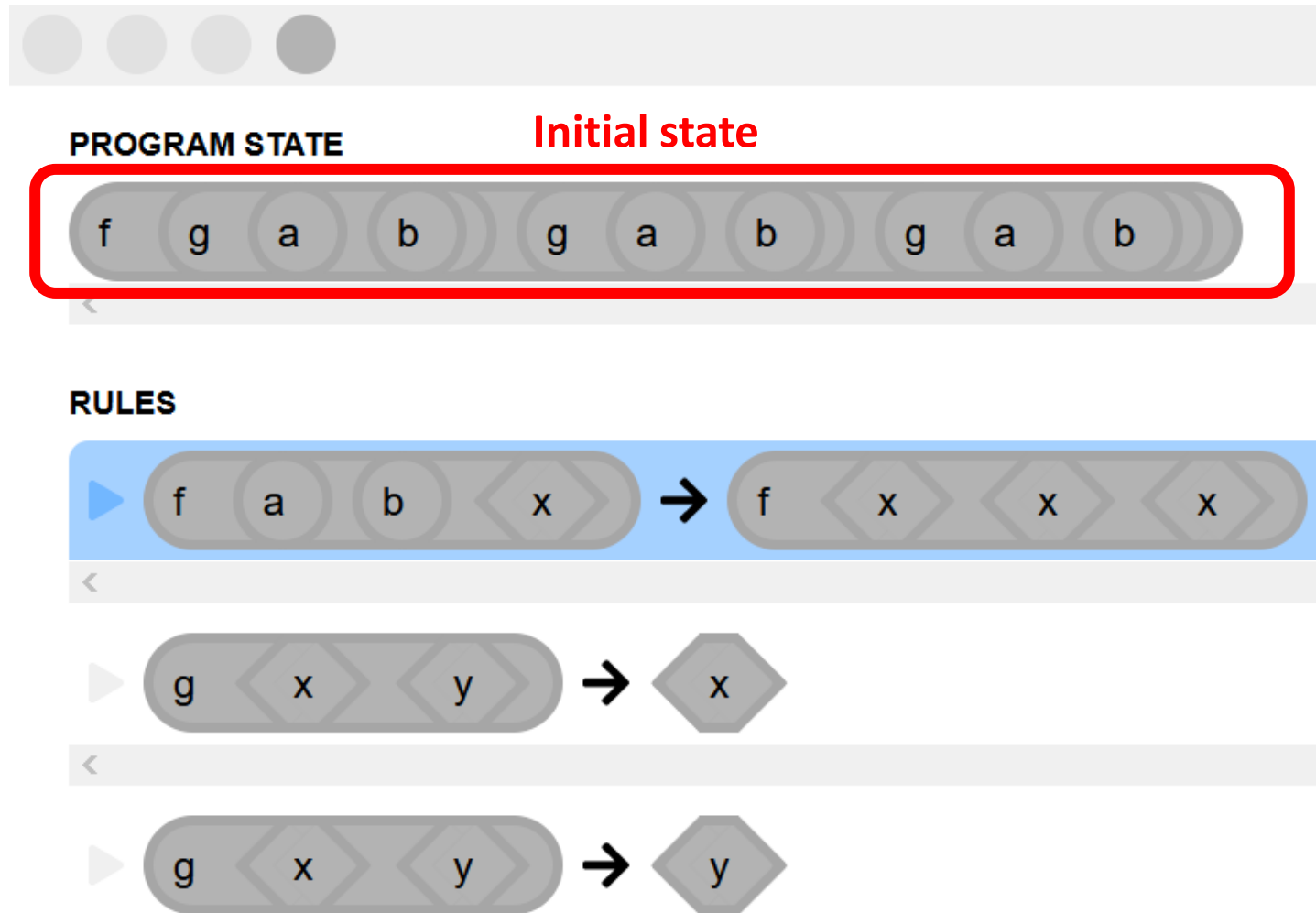
Completeness

Correctness



Undecidable in general

Termination



Reduction order

Monotone



$s_i > t \rightarrow f(s_1, \dots, s_i, \dots, s_n) > f(s_1, \dots, t, \dots, s_n)$
For f of arity n

Close under substitution



$s > t \rightarrow \sigma s > \sigma t$, for all substitution σ

Well-founded



no infinite descending chain

$(\mathbb{N}, <)$ is well-founded

$(\mathbb{Z}, <)$ is not well-founded

Termination

A term rewriting system is **terminating**

if and only if

it admits a **compatible reduction order** $<$
(if $l > r$ for every rewrite rule $l \rightarrow r$)



Verification of termination

Polynomial interpretation

$$f(a, x) \rightarrow x$$

$$f(g(x), y) \rightarrow g(f(x, y))$$

weight
→

$$w(a) = 1$$

$$w(g(t)) = 1 + w(t)$$

$$w(f(t_1, t_2)) = 2w(t_1) + w(t_2)$$

Polynomial interpretation

$$f(a, x) \rightarrow x$$

$$w(f(a, x)) = 2 + w(x)$$

$$w(x) = w(x)$$

$$w(f(a, x)) > w(x)$$

$$f(g(x), y) \rightarrow g(f(x, y))$$

$$w(f(g(x), y)) = 2 + 2w(x) + w(y)$$

$$w(g(f(x, y))) = 1 + 2w(x) + w(y)$$

$$w(f(g(x), y)) > w(g(f(x, y)))$$

Reduction order \rightarrow Termination

Algorithms

Recursive Path Ordering



Order based on the mutisets

Knuth-Bendix Ordering



Based on weights assigned to operators

Dependency pairs



Prove innermost termination

Termination

APROVE

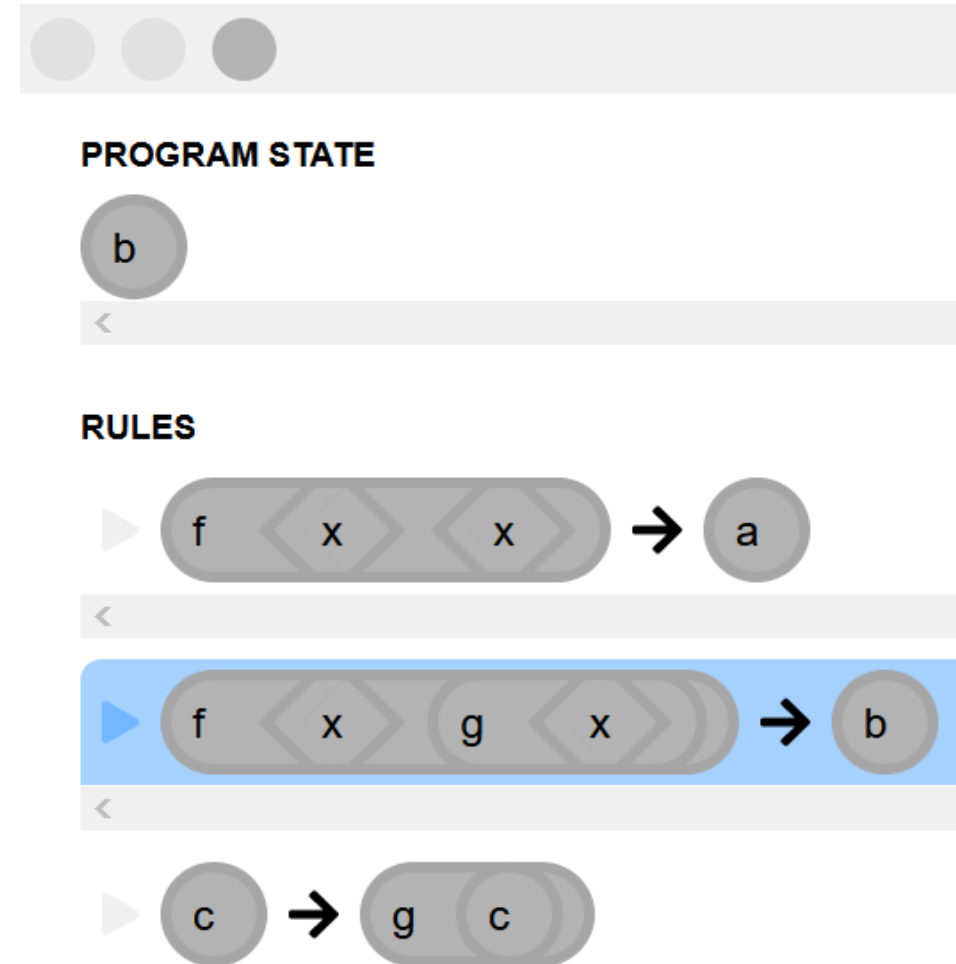
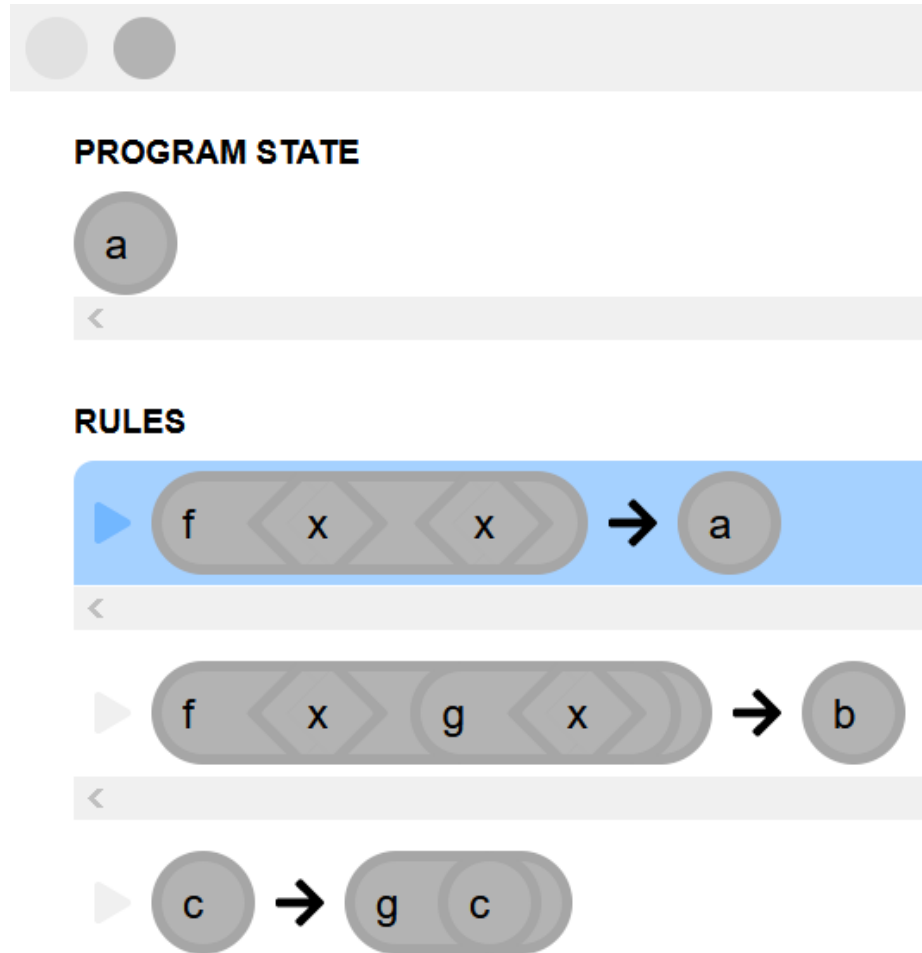
=

Direct proof
(polynomial, LBO, KBO,...)

+

Dependency pairs and
size-change principle

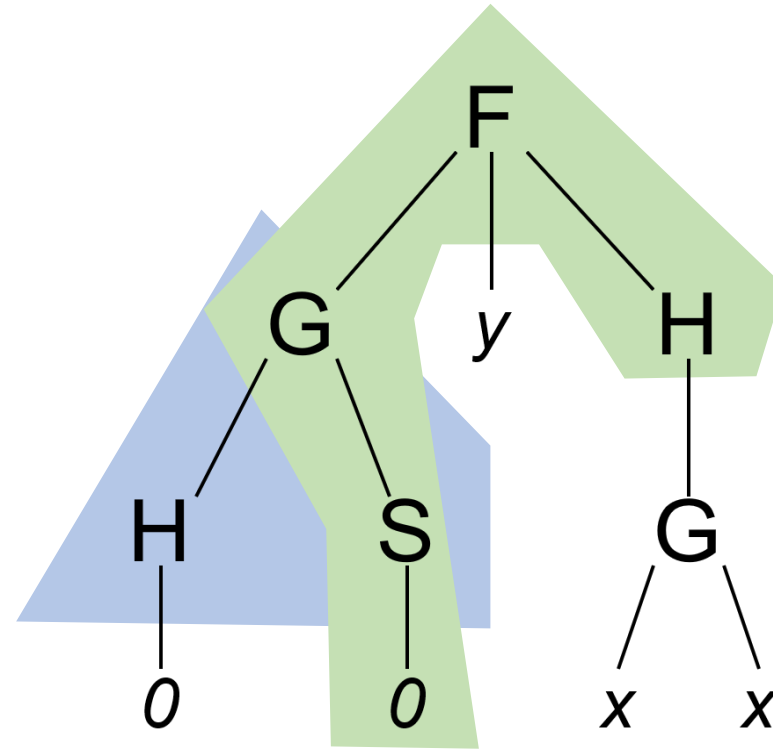
Confluence



Overlap and critical pairs

$$\rho_1 : F(G(x, S(0)), y, H(z)) \rightarrow x$$

$$\rho_2 : G(H(x), S(y)) \rightarrow y$$



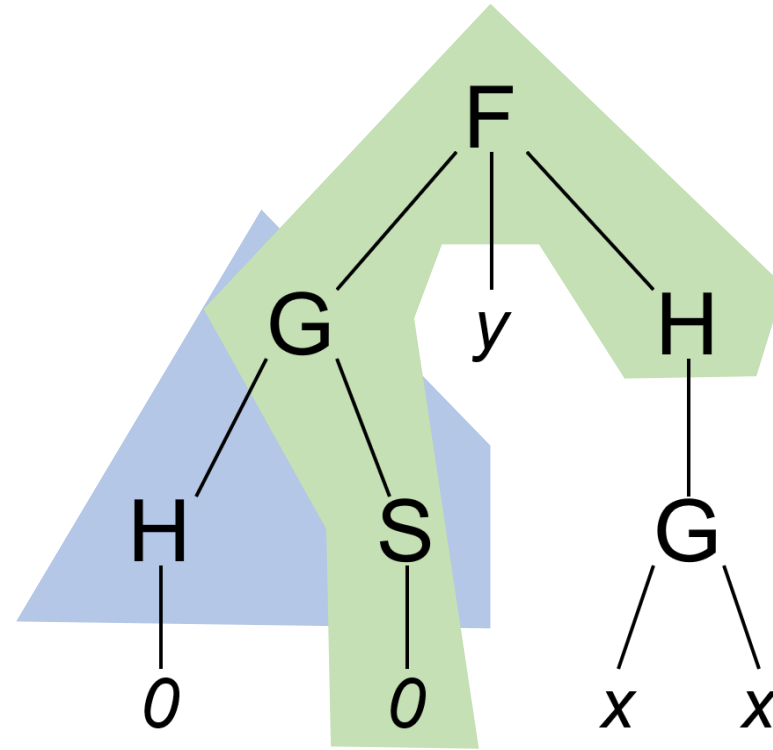
Overlap and critical pairs

Overlapping:

Term: $F(G(H(0), S(0)), y, H(z))$

$F(G(\square, S(0)), \square, H(\square))$

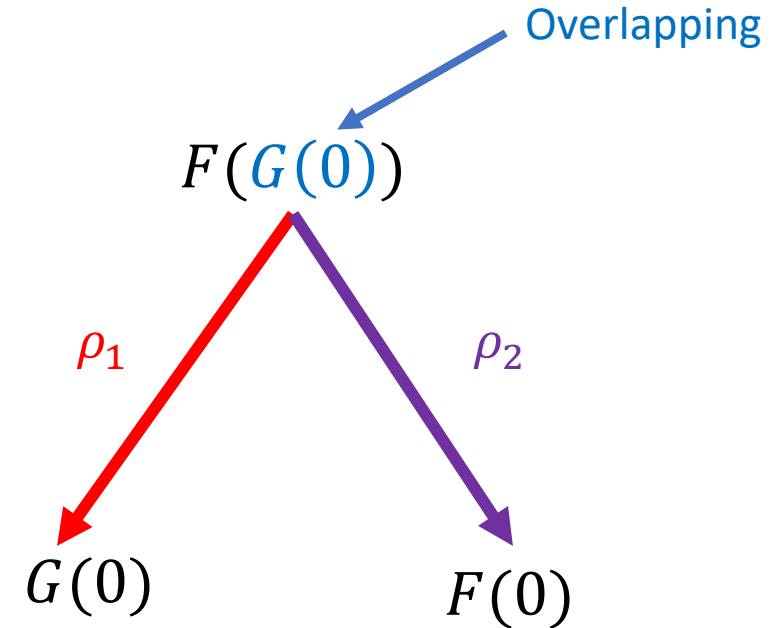
$G(H(\square), S(\square))$



Overlap and critical pairs

$$\rho_1 : F(x) \rightarrow G(0)$$

$$\rho_2 : G(x) \rightarrow 0$$



$\langle G(x), F(x) \rangle$ is called **critical pair**

Critical Pair Lemma

A terminating rewriting system is confluent

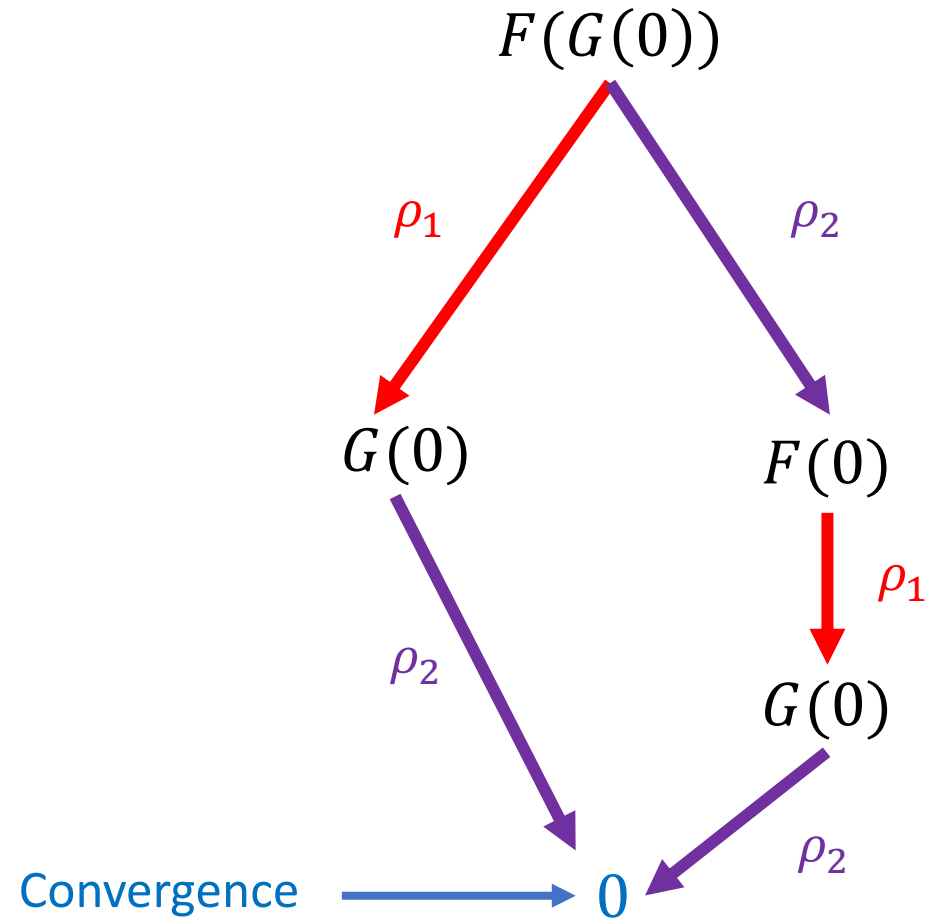
if and only if

all critical pairs are convergent

Critical Pair Lemma

$$\rho_1 : F(x) \rightarrow G(0)$$

$$\rho_2 : G(x) \rightarrow 0$$



Knuth-Bendix completion

Input:

A set of equation

A reduction ordering $<$

$$1 \cdot x = x$$

$$x^{-1} \cdot x = 1$$

$$(x \cdot y) \cdot z = x \cdot (y \cdot z)$$

Non-confluent

Knuth-Bendix completion

Output:

Terminate successfully



Terminating and confluent rewrite system

Loop indefinitely



Non-terminating rewrite system

Fail



Rule which cannot be ordered
(i.e. commutative operator)

Knuth-Bendix completion

Basic rules:

Orienting



Transform $s = t$ to $s \rightarrow t$

Adding



Add $s = t$ in the set of equation

Simplifying



Simplify $s = t$ in $s' = t'$

Deleting



Delete trivial rules $s = s$

Knuth-Bendix completion

Adding



Add $s = t$ in the set of equation

$$\begin{array}{lcl} (x * y) * z & \rightarrow & x * (y * z) \\ x * x & \rightarrow & x \end{array}$$



$$\begin{array}{c} (x * x) * z \\ \swarrow \quad \searrow \\ x * z \quad x * (x * z) \end{array}$$

No convergent

$$\begin{array}{lcl} (x * y) * z & \rightarrow & x * (y * z) \\ x * x & \rightarrow & x \\ x * (x * z) & \rightarrow & x * z \end{array}$$

New rule added

Knuth-Bendix completion

Completion process:

1. For each equation $s = t$ reduce s and t to normal form s' and t'
2. Fill the set of rules using basic operators and reduction ordering
3. If the algorithm terminate successfully: terminating and confluent rewrite system

Knuth-Bendix completion

Completion for axioms of groups:

$$\begin{aligned}1 \cdot x &= x \\ x^{-1} \cdot x &= 1 \\ (x \cdot y) \cdot z &= x \cdot (y \cdot z)\end{aligned}$$



$$\begin{aligned}1 \cdot x &\rightarrow x \\ x^{-1} \cdot x &\rightarrow 1 \\ (x \cdot y) \cdot z &\rightarrow x \cdot (y \cdot z) \\ x^{-1} \cdot (x \cdot y) &\rightarrow y \\ 1^{-1} &\rightarrow 1 \\ x \cdot 1 &\rightarrow x \\ (x^{-1})^{-1} &\rightarrow x \\ x \cdot x^{-1} &\rightarrow 1 \\ x \cdot (x^{-1} \cdot y) &\rightarrow y \\ (x \cdot y)^{-1} &\rightarrow y^{-1} x^{-1}\end{aligned}$$

Educational tools

$T_T T_2 / \text{RS}^{\text{TOOL}}$

Termination

CSI

Confluence

KBCV

Completion

TRS tool

Parcourir... Aucun fichier sélectionné.

Upload

```
(VAR x y)
(RULES
  f(x,y) -> x
  f(x,y) -> f(x,g(y))
  g(x) -> h(x)
  F(g(x),x) -> F(x,g(x))
  F(h(x),x) -> F(x,h(x))
)
(COMMENT Example 6 of \cite{AT97})
(COMMENT %% TagRevision: 1 %%)
(COMMENT %% Tags: [4ec3f85c01836]non_left_linear{};[4ec3f87f0f1e0]r
```

Go!

50



Rewrites Limit (Use with caution)

TRS tool

$R_0 = f(x, y) \rightarrow x$
R_0 is Left-Linear
R_0 is Right-Linear
R_0 is Linear
R_0 is Collapsing
R_0 is not Duplicating
R_0 is not Conservative
R_0 is Destructive

TRS
The TRS is not Left-Linear
The TRS is not Right-Linear
The TRS is not Linear
The TRS is Collapsing
The TRS is not Duplicating
The TRS is not Conservative
The TRS is Destructive
The TRS is not Orthogonal
The TRS is not Almost Orthogonal
The TRS is not Weakly Orthogonal
The TRS is Locally Confluent
Unknown confluence for The TRS
The TRS is non terminating Infinite Loop: $f(x, g(y)) \rightarrow \underline{f(x, g(g(y)))}$

TTT2

Tyrolean Termination Tool 2 (1.20)

1. Input Term Rewrite System

For input use the **standard TRS format**.

select example ...

or upload file

Parcourir...

Aucun fichier sélectionné.

```
(VAR x y)
(RULES
  add(0,y) -> y
  add(s(x),y) -> s(add(x,y))
  mul(0,y) -> 0
  mul(s(x),y) -> add(y,mul(x,y))
)
```

2. Select Strategy

☒ FAST ☐ FBI ☐ HYDRA ☐ LPO ☐ KBO ☐ POLY ☐ MAT(2) ☐ MAT(3) ☐ COMP ☐ COMPLEXITY
☐ EXPERT

3. Encode State into URL (optional)

encode URL clear URL

4. Start TTT2

check ☐ use HTML output if available (*experimental feature*)

CSI

Enter a **TRS** or **HRS** or upload a file [browse...](#)

Examples:

[trs1](#)[trs2](#)[hrs1](#)[hrs2](#)[CSI 0.1](#)[CSI 0.6](#)[CSI 1.1](#)[CSI 1.1✓](#)[UNR](#)[UNC](#)[CR](#)[CSI 1.2.4](#)[CSI 1.2.4✓](#)[CSI^ho](#)[encode URL](#)[clear URL](#)[reset](#)[submit](#)

KBCV

Orient →

Orient ←

Simplify

Delete

Compose

Collapse

Deduce

Completion

Undo

Redo

Equations

from 1 to 500

1. $f(f(x, y), z) \approx f(x, f(y, z))$

2. $f(x, c) \approx x$

3. $f(x, g(x)) \approx c$

Rules

from 1 to 500

LPO Precedence

Undo / Redo Stack

1. start : Welcome to the 'Knuth-Bendix Completion Visualizer'!

2. add : equations $f(f(x,y),z)=f(x,f(y,z))$, $f(x,c)=x$, $f(x,g(x))=c$ were added

Status Messages

Welcome to the 'Knuth-Bendix Completion Visualizer'!

equations $f(f(x,y),z)=f(x,f(y,z))$, $f(x,c)=x$, $f(x,g(x))=c$ were added

file '.kbcvinit' loaded!

Performance tools

Termination

MU-TERM

NaTT

APROVE

Confluence

ACP

Saigawa

Completion

Maxcomp

Proof verification

CoLoR

CeTA

Hybrid tools

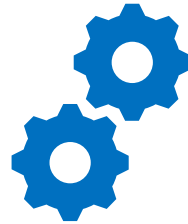


MoudE3

CiME



Rewriting toolkit

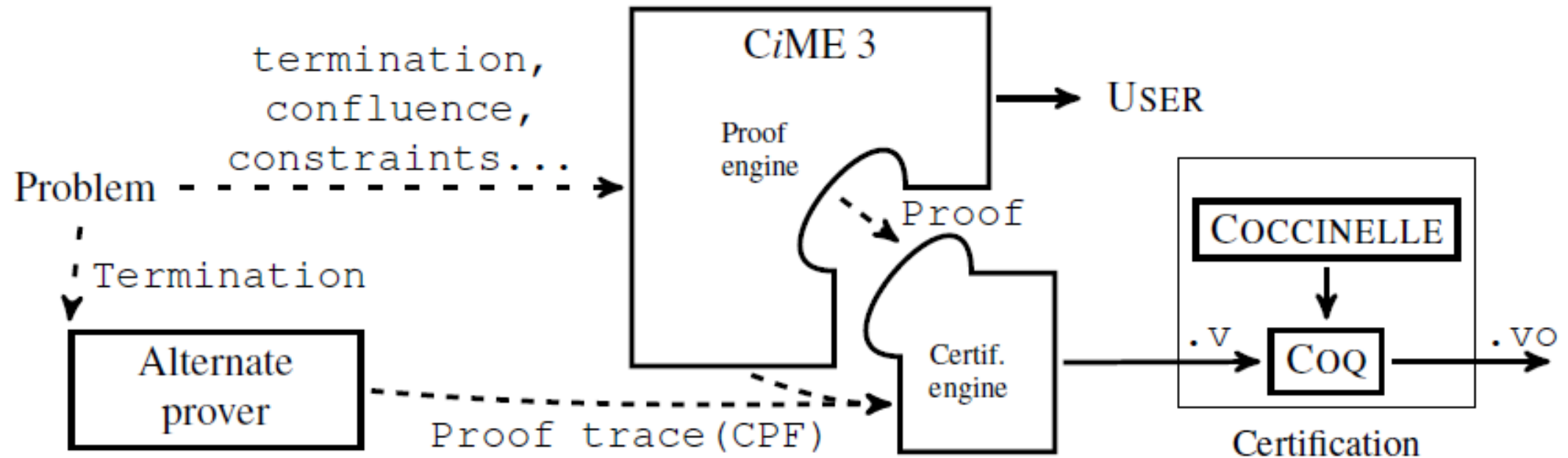


Proof engine



Proof certification

CiME



CiME

Examples of declarations

```
let X = variables "x,y";  
  
let F = signature "plus : binary; 0:constant; S:unary;";  
  
let T = algebra F;  
  
let t1 = term T "S(0)";  
  
let R = trs T "plus(0,x) -> x; plus(S x, y) -> S(plus(x,y));";  
  
let c = order_constraints T "0 < S(0) /\ S(plus(x,y)) < plus(S(x),y)";
```

CiME

Definition of signatures

```
CiME> let F_peano = signature "  
    0 : constant; s : unary; +,* : infix binary;  
    ";
```

```
F_peano : signature = signature "* : 2; s : 1; + : 2; 0 : 0"
```

```
CiME>let X = variables "x,y,z";
```

```
X : variable_set = variables "z,x,y"
```


CiME

Definition of algebra and terms

```
CiME> let A_peano = algebra F_peano ;  
A_peano : F_peano algebra = algebra F_peano
```

```
CiME> let t = term A_peano "s(s(s(0)))*(s(0)+s(s(0)))";  
t : F_peano term = s(s(s(0))) * (s(0)+s(s(0)))
```

CiME

Term rewriting system

```
CiME> let R_peano = trs A_peano "  
  x+0 -> x;  
  x+s(y) -> s(x+y);  
  x*0 -> 0;  
  x*s(y) -> (x*y)+x;  
  ";  
  
R_peano : F_peano trs = trs A_peano "  
  x+0 -> x;  
  x+s(y) -> s(x+y);  
  x *0 -> 0;  
  x *s(y) -> (x *y)+x "
```

```
CiME> termination R_peano;  
  
CiME> coq_certify_proof R_peano;  
  
CiME> convergence R_peano ;  
  
...
```

Maude

MoudE3



Simplicity



Expressiveness



Performance

Maude

```
1  fmod BASIC-NAT is
2      sort Nat .
3
4      op 0 : -> Nat .
5      op s : Nat -> Nat .
6      op _+_ : Nat Nat -> Nat .
7
8      vars N M : Nat .
9
10     eq 0 + N = N .
11     eq s(M) + N = s(M + N) .
12 endfm
```

Maude

```

5      fmod FACTORIAL is
6      protecting NAT .
7      op _! : Nat -> NzNat .
8      var N : Nat .
9      eq 0 ! = 1 .
10     eq (s N) ! = (s N) * N ! .
11     endfm

```

```
> load factorial.mauve
> red 100 ! .
Reduce in FACTORIAL : 100 ! .
rewrites: 201 in 0ms cpu (0ms real) (~ rewrites/second)
result NzNAT:
9332621544394415268169923885626670049071596826438162146
8592963895217599993229915608941463976156518286253697920
8272237582511852109168640000000000000000000000000000
```

Maude

```
7      mod VENDING-MACHINE is
8        including VENDING-MACHINE-SIGNATURE .
9        var M : Marking .
10       rl [add-q] : M => M q .
11       rl [add-$] : M => M $ .
12       rl [buy-c] : $ => c .
13       rl [buy-a] : $ => a q .
14       rl [change] : q q q q => $ .
15     endm
```

Maude

Inductive Theorem Prover (ITP)

Sufficient Completeness Checker (SCC)

Church-Rosser Checker (CRC)

Coherence Checker (ChC)

Maude Termination Tool (MTT)

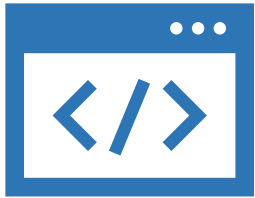


Maude Formal Environment (MFE)

Tools overview

	Maude	CiME
Extensibility	+	≈
Still active	≈	—
I/O files	+	+
Syntax	+	+
Documentation	+	—

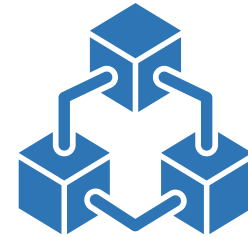
Maude MSOS Tool (MMT)



Language specification

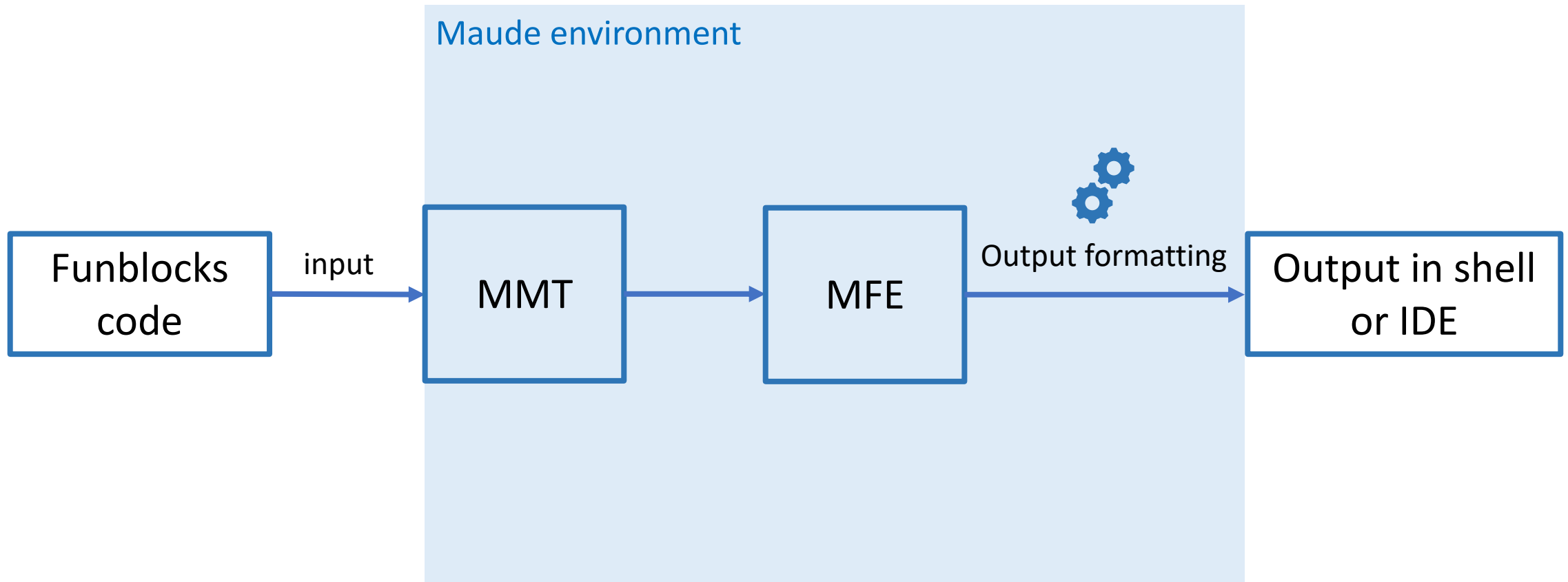


Based on transition rules



Modularity

System diagram



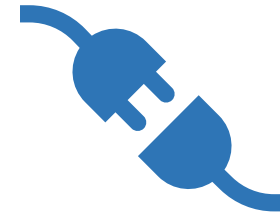
I/O in Maude



Standard input/output



File handling



Sockets

I/O in Maude

```
7      mod VENDING-MACHINE is
8        including VENDING-MACHINE-SIGNATURE .
9        var M : Marking .
10       rl [add-q] : M => M q .
11       rl [add-$] : M => M $ .
12       rl [buy-c] : $ => c .
13       rl [buy-a] : $ => a q .
14       rl [change] : q q q q => $ .
15     endm
```

I/O in Maude

```
5   fmod VENDING-MACHINE-SIGNATURE is
6     sorts Coin Item Marking .
7     subsorts Coin Item < Marking .
8     op __ : Marking Marking -> Marking [assoc comm id: null] .
9     op null : -> Marking .
10    op $ : -> Coin [format (r! o)] .
11    op q : -> Coin [format (r! o)] .
12    op a : -> Item [format (b! o)] .
13    op c : -> Item [format (b! o)] .
14  endfm
```

I/O in Maude

```
5    load vending-machine-signature.maude
6
7    fmod VENDING-MACHINE-GRAMMAR is
8      protecting VENDING-MACHINE-SIGNATURE .
9      protecting NAT .
10     sort Action .
11     op insert $ : -> Action .
12     op insert q : -> Action .
13     op show basket : -> Action .
14     op show credit : -> Action .
15     op buy__(s) : Nat Item -> Action .
16   endfm
```

I/O in Maude

```
5  load vending-machine-grammar.maude
6  load buying-strats.maude
7  load file.maude
8
9  mod VENDING-MACHINE-IO is
10      .
11      .
12      .
95  rl < 0 : X | action : insert $, marking : M, Atts >
96  => < 0 : X | action : idle,
97      marking : downTerm(insertCoin('add-$, upTerm(M)), null), Atts >
98  write(stdout, 0, "one dollar introduced\n") .
```

I/O in Maude

Enter in external environment

```
[Maude> erew vending machine .  
erewrite in VENDING-MACHINE-IO : vending machine .
```

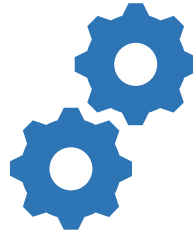
Input	> insert \$
Output	one dollar introduced
	> buy 1 a(s)
	1 apples bought

Parse input?

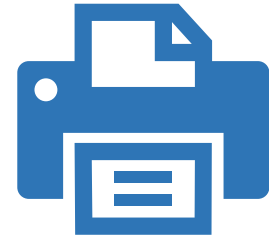
META-LEVEL module



Parse input

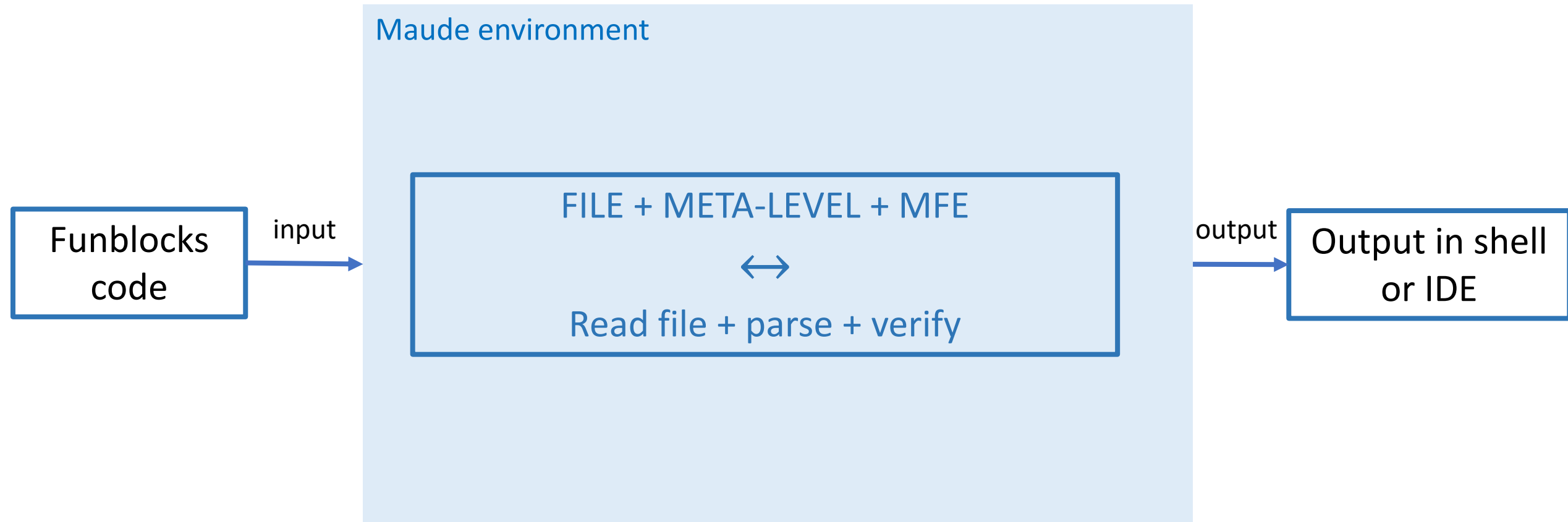


Execute

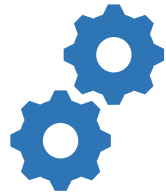


Print

System diagram



What's next?



META-LEVEL module



Handle FunBlock file input

References

1. Didier Buchs, modelisation verification course material, 2018
2. Dimitri Racordon, Emmanouela Stachtiri, Damien Morard, Didier Buchs, Functional Block Programming and Debugging, 2020
3. Nachum Dershowitz, Jean-Pierre Jouannaud, Rewrite Systems, 1990
4. Nachum Dershowitz, Computing with Rewrite Systems, 1985
5. Terese, Term Rewriting Systems, 2003
6. Thomas Sternagel and Harald Zankl, KBCV-Knuth-Bendix Completion Visualizer, 2012
7. Thomas Artsa, Jürgen Giesl, Termination of term rewriting using dependency pairs, 2000

References

8. Jürgen Giesl, René Thiemann, Peter Schneider-Kamp, Stephan Falke, Automated Termination Proofs with AProVE, 2004
9. D. Kapur, P. Narendran, Path ordering for proving termination of term rewriting systems, 1985
10. Jeremy Dick, John Kalmus and Ursula Martin, Automating the Knuth Bendix ordering, 1990
11. E. Contejean, P. Courtieu, J. Forest, O. Pons, X. Urbain, Automated Certified Proofs with CiME3, 2011
12. F. Chalub, C. Braga, Maude MSOS Tool, 2005
13. S. Winkler, A. Middeldorp, Tools in Term Rewriting for Education, 2020
14. A. Salvador, L. Salvador, Term Rewriting Systems .Net Framework, 2013

References (links)

Database of Rewriting Systems

- <http://rewriting.loria.fr/systems.html>
- <http://www.jaist.ac.jp/~hirokawa/tool/>

Knuth-Bendix Completion Visualizer

- <http://cl-informatik.uibk.ac.at/software/kbcv/>

Knuth-Bendix Completion subject-based thesis

- <https://homepage.divms.uiowa.edu/~astump/papers/thesis-wehrman.pdf>

References (links)

Prolog implementation of the Knuth-Bendix completion procedure

- <https://www.metalevel.at/trs/>

Maude tools

- <http://maude.lcc.uma.es/CRChC/>
- <http://www.lcc.uma.es/%7Eduran/MTT/>
- <http://maude.sip.ucm.es/debugging/>

Wikipedia

- [https://fr.wikipedia.org/wiki/Compl%C3%A9tion de Knuth-Bendix](https://fr.wikipedia.org/wiki/Compl%C3%A9tion_de_Knuth-Bendix)
- [https://fr.wikipedia.org/wiki/Paire critique](https://fr.wikipedia.org/wiki/Paire_critique)

References (links)

TRS tool:

- <http://tfmserver.dsic.upv.es:8080/Home.html>

Make FunBlocks alive

Marvin FOURASTIE

Master project