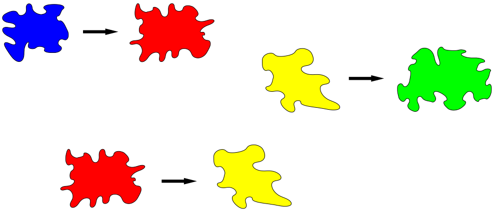
Efficient Rewriting Techniques

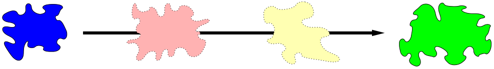
Muck van Weerdenburg

Rewriting - What is it?

Collection of rules



that describe how we can manipulate objects.

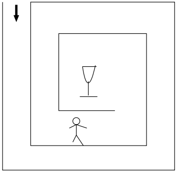
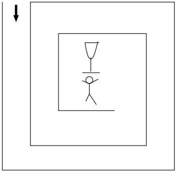


Rewriting - What is the use?

We can give rules such that we can calculate things like:

$$(3^2 + 20) * 13 \quad \rightarrow \ldots \rightarrow \quad 377$$

or

up(right()) $\rightarrow \ldots \rightarrow$ 

Goal - What do we want?

We often have very big or very many calculations.

E.g. analysing communication protocol for mobile phone easily requires billions of rewrites.

We want speed!

Goal - What did we do?

- Formal definition of *match trees*
- Given a method for efficient *term construction*
- New strategy framework: *strategy trees*

Match Trees - Matching?

Matching is the process of seeing if a rule can be applied:

Can $0 * x \rightarrow 0$ be applied to $0 * (1 + 1)$?

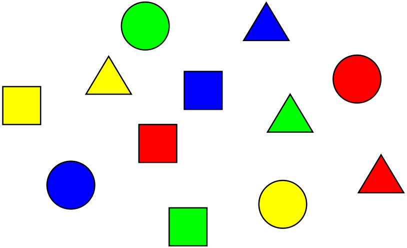
Yes, take $(1 + 1)$ for x.

Can $0 * x \rightarrow 0$ be applied to $1 * (1 + 1)$?

No, $1 * (1 + 1)$ does not start with 0.

Match Trees - Example

Which of the following objects matches  ?



Match Trees - The naive approach

Is it red and round? No.

Is it green and round? No.

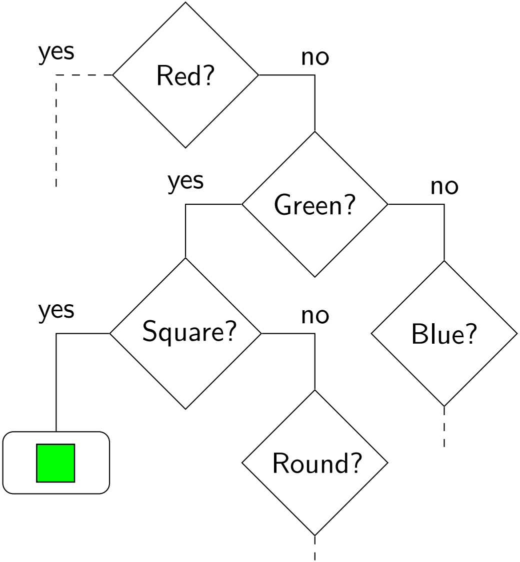
. . .

Is it red and square? No.

Is it green and square? Yes: 

. . .

Match Trees - Using a match tree



Match Trees - The difference

Naive: on average 12 questions needed ($\approx$ shapes times colours)

Trees: on average 4.5 questions needed ($\approx$ shapes plus colours)

Term Construction

We have given a method to construct terms and

- add function annotations to mark already rewritten parts
- add function annotations to avoid unrewritable parts
- directly rewrites parts that will be rewritten later on any way

This method avoids a lot of work

(E.g. trying to rewrite terms that cannot or are already rewritten)

Term Construction

Example: assume that we apply rule $\sin(x + x) \to \sin(2 * x)$ to $\sin(3 + 3)$.

Note that 3 is already rewritten.

"*add function annotations to mark already rewritten parts*"

This gives $\sin(2 * 3)$

(Or actually $\sin(2 *^{\{2\}} 3)$)

Term Construction

Example: assume that we apply rule $\sin(x + x) \rightarrow \sin(2 * x)$ to $\sin(3 + 3)$.

Note that 3 is already rewritten.

"*add function annotations to avoid unrewritable parts*"

We have no rewrite rules for 2

This gives $\sin(2 * 3)$

Term Construction

Example: assume that we apply rule $\sin(x + x) \to \sin(2 * x)$ to $\sin(3 + 3)$.

Note that 3 is already rewritten.

"*directly rewrites parts that will be rewritten later on anyway*"

We first rewrite $2 * 3$ to 6 as we will need it later on anyway

Then we get $\sin(6)$

Strategy Trees - What is a strategy?

A strategy defines how to make choices.

$$(0+0)*(1+1) \quad \rightarrow \quad 0*(1+1)$$

or

$$(0+0)*(1+1) \quad \rightarrow \quad (0+0)*2$$

Strategy Trees - What does it matter?

Often used simple strategy: *innermost*

$$(0+0)*(1+1) \quad \rightarrow \quad 0*(1+1) \quad \rightarrow \quad 0*2 \quad \rightarrow \quad 0$$

Better: *just-in-time*

$$(0+0)*(1+1) \quad \rightarrow \quad 0*(1+1) \quad \rightarrow \quad 0$$

Strategy Trees - Better than just-in-time?

Numbers within something (e.g. sets, lists, boxes, trains)

With just-in-time:

$$\text{boxed?}(\text{box}(1+1)) \;\rightarrow\; \text{boxed?}(\boxed{1+1}) \;\rightarrow\; \text{boxed?}(\boxed{2}) \;\rightarrow\; \text{yes}$$

Quicker: *strategy trees*

$$\text{boxed?}(\text{box}(1+1)) \;\rightarrow\; \text{boxed?}(\boxed{1+1}) \;\rightarrow\; \qquad\qquad \text{yes}$$

Evaluation

We have evaluated each of the mentioned techniques

All test have shown a positive impact on rewriting