

New Developments in SPASS+T

Uwe Waldmann

MPI für Informatik, Saarbrücken, Germany

joint work with Michel Ludwig and Stephan Zimmer

SPASS+T

SPASS [Weidenbach et al.]:

Saturation prover based on the superposition calculus
[Bachmair/Ganzinger].

Saturation proving:

Take a set of input formulas.

Generate new formulas according to some inference system
and add them to the current set.

Possibly discard useless formulas.

Stop if FALSE is derived.

SPASS+T

SPASS+T uses three complementary methods to add arithmetic knowledge to SPASS:

- external SMT procedure (e. g., Yices, CVC, ...),

- standard axioms,

- built-in arithmetic simplification and inference rules.

Note: Pragmatic approach, not striving for completeness.

Hierarchic Combinations of Provers

Coupling SPASS and an SMT procedure:

SPASS saturates the set of input formulas as usual.

A control module collects all ground formulas derived by SPASS and calls the SMT procedure repeatedly with the formulas collected so far.

As soon as one of the provers detects a contradiction: stop.

Standard Axioms

The external SMT procedure is supplemented by a set of standard axioms, e. g.:

$$\forall x, y. ((x + (-y)) + y = x),$$

$$\forall x. (x + 0 = x),$$

$$\forall x. (x < x + 1).$$

Built-in Arithmetic

In addition, SPASS+T uses a number of built-in arithmetic simplification techniques, e. g.:

$$\frac{C[c_1 + c_2]}{C[c_0]} \quad \text{where } c_0 = c_1 + c_2$$

$$\frac{C \vee [\neg] t + c_1 \sim c_2}{C \vee [\neg] t \sim c_0} \quad \text{where } c_0 = c_2 - c_1$$

Built-in Arithmetic

Advantages of built-in arithmetic:

- Find solutions for variables:

$$\forall x. (\neg x + 2 = 6 \vee p(x)) \quad \mapsto \quad p(4).$$

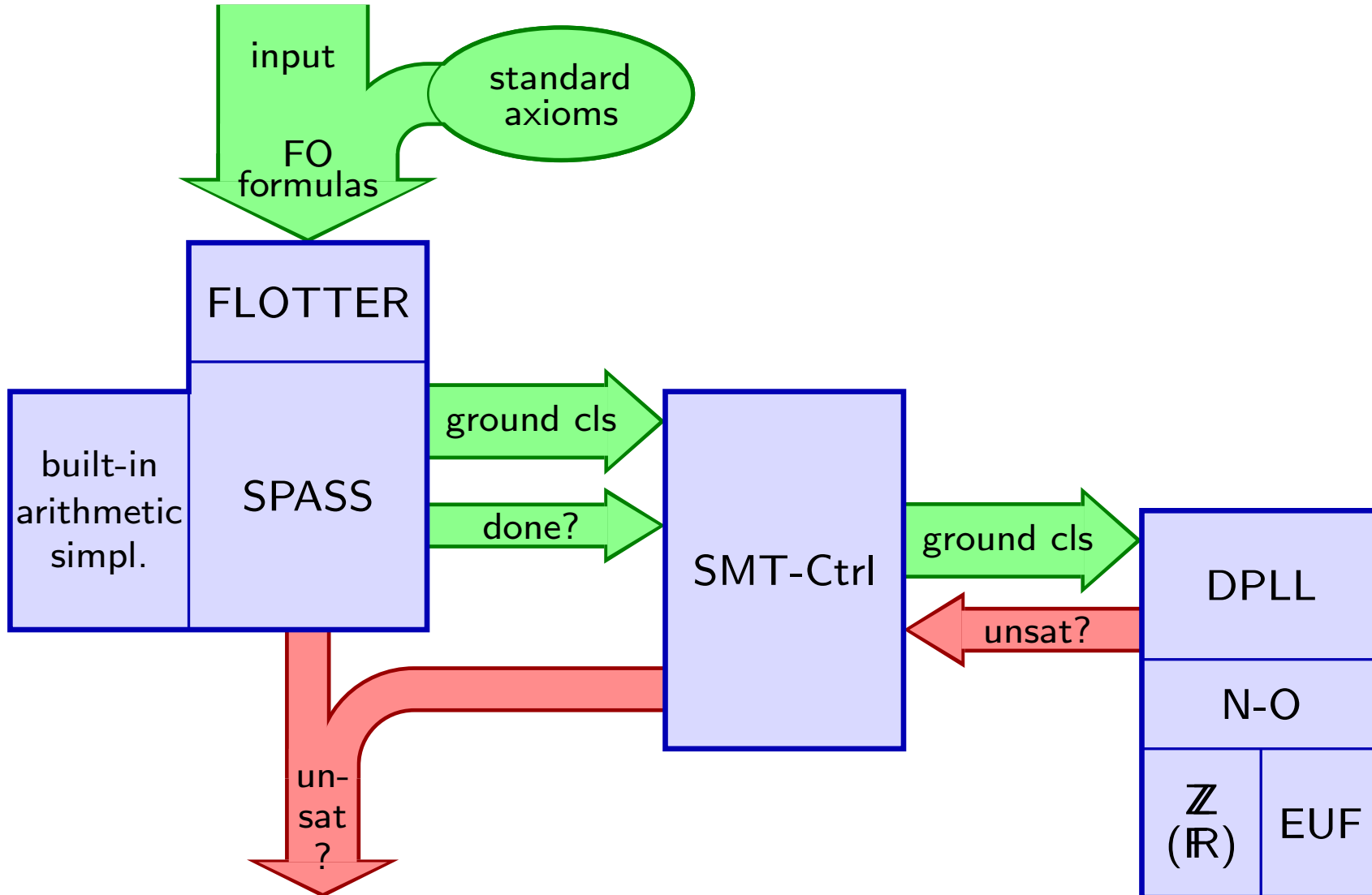
- Avoid cluttering the search space:

$$p(2 + 1), p(1 + 2), p(1 + (1 + 1)) \quad \mapsto \quad p(3).$$

- Make equations like $\forall x. (x + 0 = x)$ applicable:

$$p(((a + 2) - 1) - 1) \quad \mapsto \quad p(a + 0) \quad \mapsto \quad p(a).$$

SPASS+T



What's new?

Changes since 2006:

- splitting support

- ordered chaining inference rule

- new term ordering

- improved number representation

- proof documentation for SMT subproofs

- ...

Note:

- currently two independent branches

- (to be merged during the next month).

Splitting

SPASS uses splitting to eliminate variable-disjoint positive disjunctions

$$N \cup \{A \vee B\} \rightarrow N \cup \{A\} \mid N \cup \{B\}.$$

So far: Rule must be switched off in SPASS+T, since the conclusions of splitting are not consequences of the premises (\rightarrow unsound SMT proofs).

Splitting

Solution:

Pass individual branches of the splitting tree to the SMT procedure.

If contradiction is found: Extract unsatisfiable core and use it for intelligent backtracking.

Originally planned: start only one instance of SMT procedure, assert and retract branch literals as needed; but manipulating the generated set of clauses oneself and restarting the SMT procedure turned out to be more efficient.

Proof Documentation

SMT refutations appear now in the final SPASS proof like any other inference rule application (using the unsatisfiable core).

Theory Instantiation

Resolution and superposition will not always generate ground formulas usable in the SMT procedure.

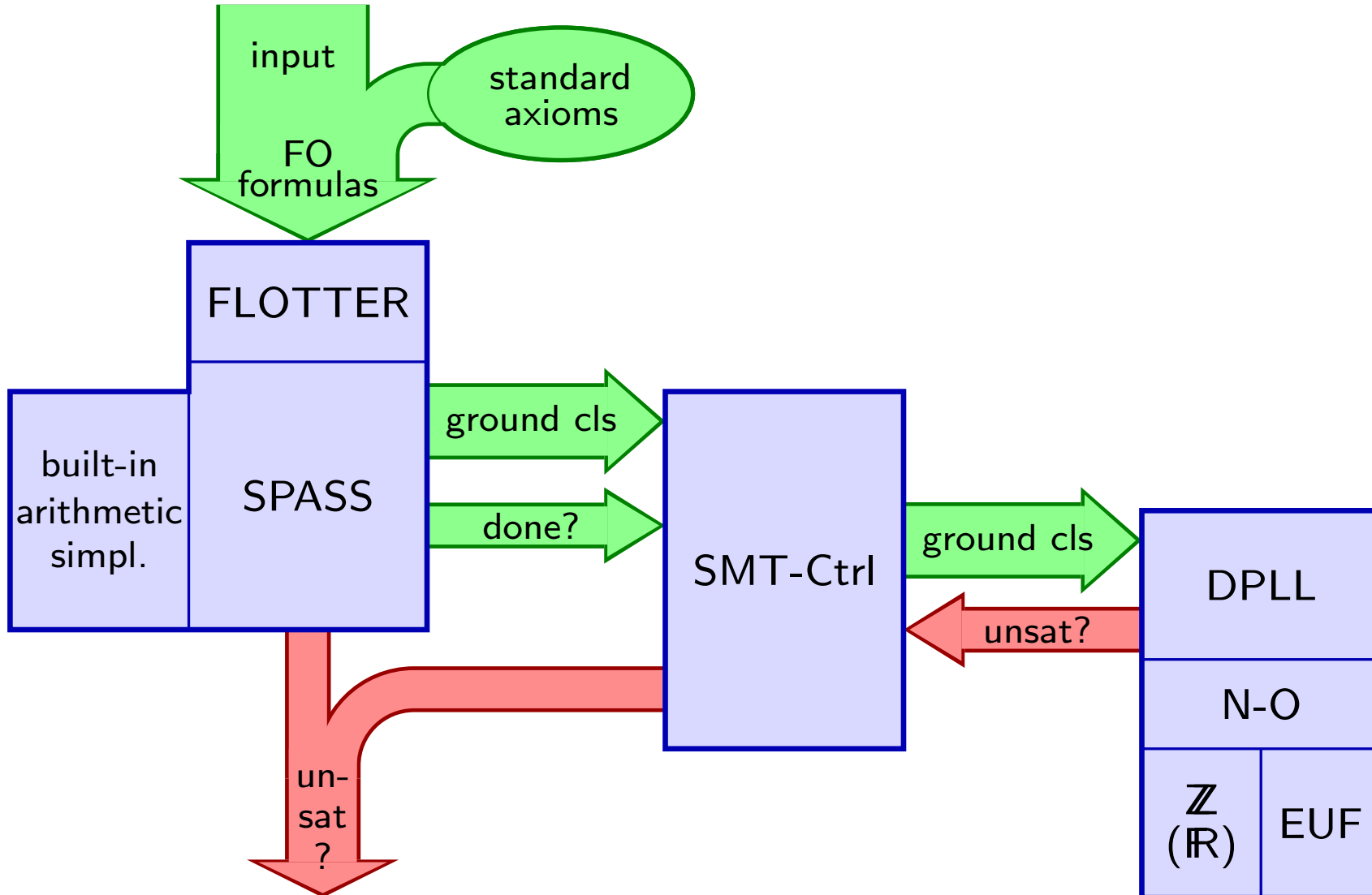
So far, SPASS+T contained an additional rule for heuristic instantiation of clauses.

Current SMT provers do this quite well; so there is no need to duplicate the work in SPASS+T.

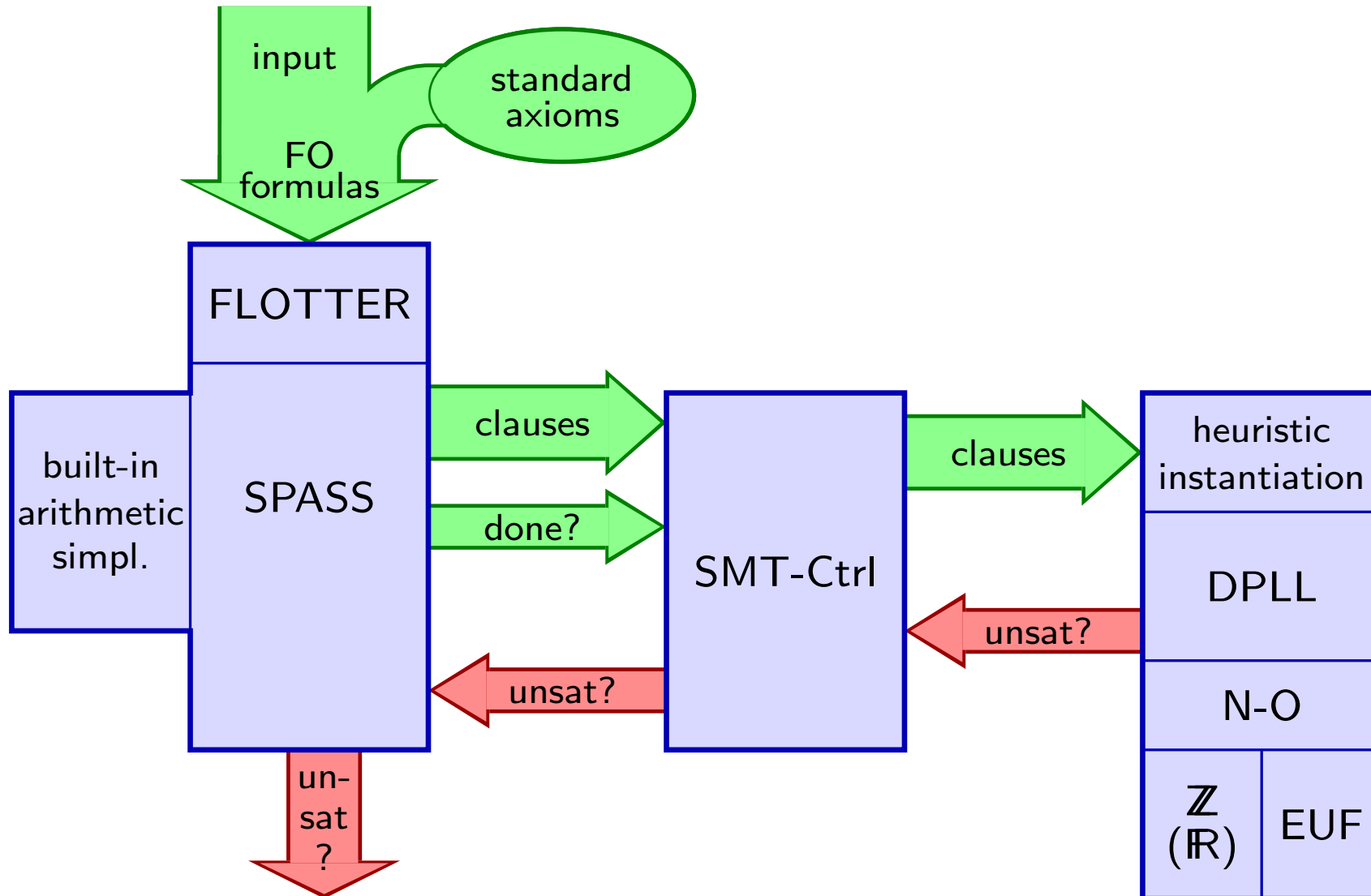
The new version of SPASS+T passes non-ground clauses to the SMT prover.

Caveat: run SMT prover with **very** limited resources.

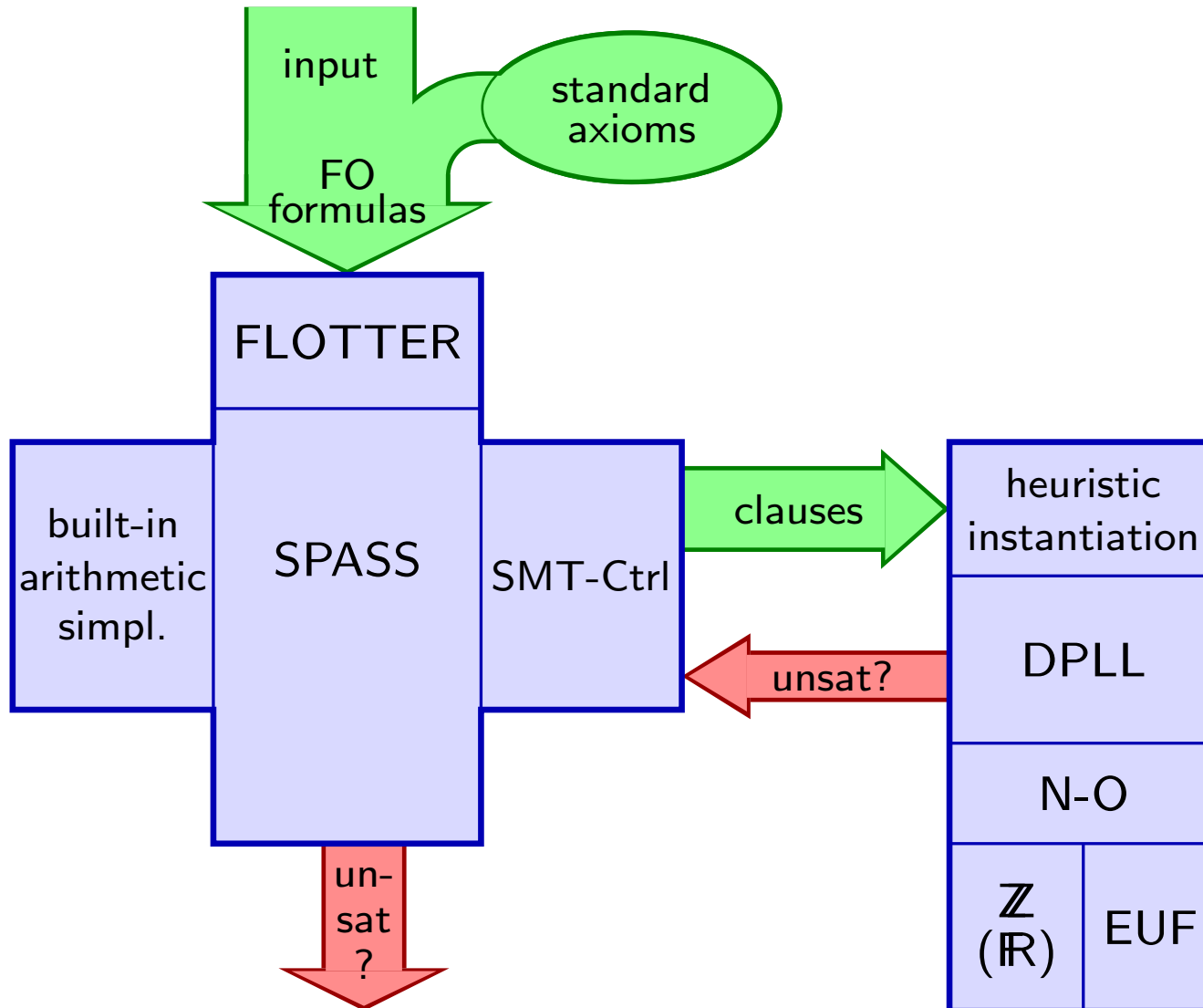
SPASS+T: Old Architecture



SPASS+T: New Architecture



SPASS+T: New Architecture



Ordered Chaining

Problem:

The transitivity axiom for $<$ is sometimes needed for finding a proof, but it creates a huge search space in a resolution prover.

In general, adding transitivity to the set of standard axioms is not practical.

Ordered Chaining

Solution: Ordered chaining.

$$\frac{D' \vee t' < t \quad C' \vee s \leq s'}{(D' \vee C' \vee t' < s')\sigma}$$

where

- σ is a most general unifier of t and s ,
- neither t nor s is a variable,
- the two ordering literals are maximal,
- s and t are maximal terms in the literals.

Ordered Chaining

Additionally: Ordered chaining elimination.

$$\frac{D' \vee t' < t \quad C' \vee s \leq s'}{(D' \vee C')\sigma}$$

where

- σ is a most general unifier of t and s and of t' and s' ,
- the two ordering literals are maximal.

Triple KBO

Requirements for term ordering:

non-arithmetic formulas should be larger than arithmetic ones,
formulas that differ only by numeric constants should be compared according to the size of numerals,
complex arithmetic expressions should be larger than the numbers to which they evaluate.

Solution:

KBO with triples as weights:

$f \rightarrow (1, 0, 0)$, $+$ $\rightarrow (0, 1, 0)$, $7 \rightarrow (0, 0, 14)$,

componentwise addition, lexicographic comparison.

What Next?

The next steps:

Merging the two SPASS+T branches.

Experiments.