Challenges for Embedded Deduction

Harald Rueß

ruess@csl.sri.com

http://www.csl.sri.com/users/ruess.

Computer Science Laboratory

SRI International

333 Ravenswood

Menlo Park, CA 94025

Embedded Deduction

- Most uses of deductive engines are embedded.

  Simulators, abstractors, typecheckers, compilers, constraint solvers, model checkers, test case generators, synthesizers, test-case generators, fault tree analyzers, . . .

- Emphasis in embedded deduction is on in-the-loop exploration and debugging.
  - Does this follow from that?
  - Suppose I retract this and assert that, does it still follow?
- Embedded deduction requires rich interfaces.

Embedded Deduction

- Most uses of deductive engines are embedded.

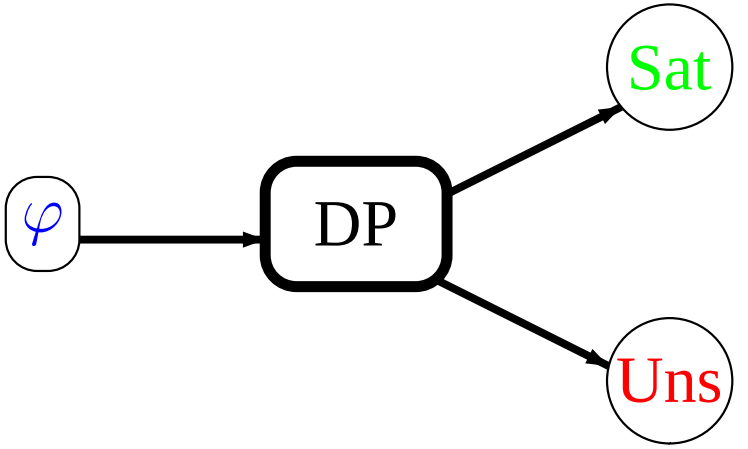
  Simulators, abstractors, typecheckers, compilers, constraint solvers, model checkers, test case generators, synthesizers, test-case generators, fault tree analyzers, . . .

- Emphasis in embedded deduction is on in-the-loop exploration and debugging.
  - Does this follow from that?
  - Suppose I retract this and assert that, does it still follow?

- Embedded deduction requires rich interfaces.

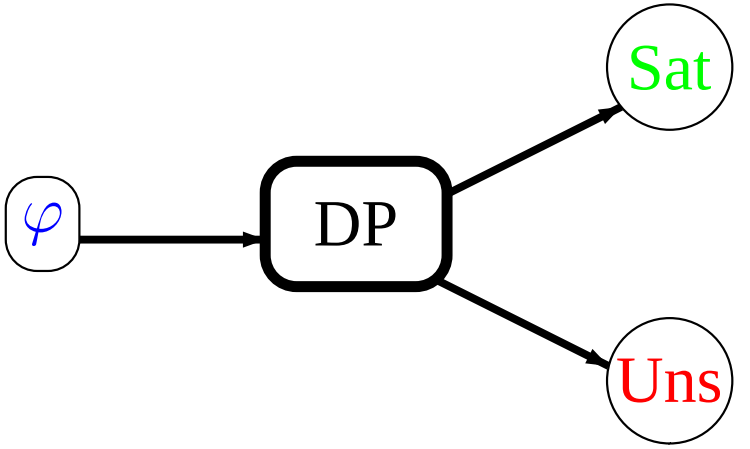
  "***The black box nature of the decision procedure is frequently destroyed by the need to integrate it***" (Boyer/Moore)

Interfaces for Embedded Deduction



- **Online.** Incremental processing of assertions and queries.
- **Resettable.** Saving, backtracking, and switching contexts.
- **Queriable.** Allow expressions to be simplified wrt a context.
- **Evidential.** Proof objects, unsatisfiable cores, and models.
- **Reliable Automation.** Prompt or even any-time response.
- **Integrable.** Fine-grained integration with other tools.

Interfaces for Embedded Deduction



- **Online.** Incremental processing of assertions and queries.
- **Resettable.** Saving, backtracking, and switching contexts.
- **Queriable.** Allow expressions to be simplified wrt a context.
- **Evidential.** Proof objects, unsatisfiable cores, and models.
- **Reliable Automation.** Prompt or even any-time response.
- **Integrable.** Fine-grained integration with other tools.

ICS (**`ics.csl.sri.com`**) tries to provide such a rich interface while keeping overhead small.

Composition in ICS

Open Inference Systems. (Ganzinger, Shankar, R.; 2004)

- ***Configurations*** | I | DP(T) |
  - Shared blackboard I consisting of shared constraints.
  - Theory-specific part DP(T) that is like a private notebook.
- ***Composition operator*** yields DP for union of theories.

I	DP(T_1)	$\otimes$	I	DP(T_2)	$\rightsquigarrow$	I	DP(T_1), DP(T_2)

- ***Refinement*** yields Nelson-Oppen and Shostak combinations.

Composition in ICS

Open Inference Systems. (Ganzinger, Shankar, R.; 2004)

- ***Configurations*** | I | DP(T) |
  - Shared blackboard I consisting of shared constraints.
  - Theory-specific part DP(T) that is like a private notebook.
- ***Composition operator*** yields DP for union of theories.

I	DP(T_1)	$\otimes$	I	DP(T_2)	$\rightsquigarrow$	I	DP(T_1), DP(T_2)

- ***Refinement*** yields Nelson-Oppen and Shostak combinations.

Lazy Integration of DP and SAT.

- ***Interface*** J. (Abstraction, Lemmas, Assignments)
- ***Composition.***

J	DP	$\otimes$	J	SAT	$\rightsquigarrow$	J	DP, SAT

- ***Optimizations.*** (unsatisfiable cores, online integration)

Summary

Where are we?

- Around $10000 - 30000$ theorems a second for DP.
- Problems with 10000s of literals using lazy DP/SAT.
- Rapid progress due to healthy competition and improved benchmarking, but "good" benchmark sets still not available.

What's next?

- $100 - 1000$ fold speed improvement over next 3 year.
- Construction and collection of better benchmark sets.
- Enough raw speed for most routine embedded applications.

Challenges!

- Practical challenges involve designing interfaces that allow flexible use without loss of efficiency.
- Designing integration architectures that mediate fine-grained interaction between inference components.