Towards Smart Proof
Search for Isabelle
PSL and all that

Yutaka Nagashima I Trustworthy System Research Group
March 2017

www.csiro.au
Example proof at Data61

```
lemma performPageTableInvocationUnmap_ccorres:
  "ccorres (K (K \<bottom>) \<currency> dc) (liftxf errstate id (K () ) ret__unsig
   (invs' and cte_wp_at' (diminished' (ArchObjectCap cap) \circ cteCap) ctS'
    and (\lambda_isPageTableCap cap))
   (UNIV \inter \lbrace ccap_relation (ArchObjectCap cap) \acute cap\rbrace[]
    (liftE (performPageTableInvocation (PageTableUnmap cap ctSlot))))
   (Call performPageTableInvocationUnmap 'proc"
apply (simp only: liftE_liftM ccorres_liftM_simp)
apply (rule ccorres_gen_asm)
apply (cinit lift: cap_' ctSlot_')
apply csymbm
```

The salary range for this position is AUD 65,000 to 90,000 for recent graduates,

```
apply (subgoal_tac "capPTMappedAddress cap
  = (\lambda cp. if to_bool (capPTIsMapped_CL cp)
    then Some (capPTMappedASID_CL cp, capPTMappedAddress:
    else None) (cap_page_table_cap_lift capa)"
apply (rule ccorres_Cond_rhs)
apply (simp add: to_bool_def)
apply (rule ccorres_rhs_accoc):
```
Towards Smart Proof Search. | Yutaka Nagashima

PSL and try-hard for Isabelle/HOL

The percentage of automatically proved obligations out of 1526 proof obligations (timeout = 300s)

- Part 1: 73%
- Part 2: 28%

Not specific to Isabelle!
Other ITPs / Logic Programming
Isabelle/HOL before PSL

proof goal  context

tactic / sub-tool

error-message

It's blatantly clear
You stupid machine, that what
I tell you is true
(Michael Norrish)
Towards Smart Proof Search. | Yutaka Nagashima

PSL (Proof Strategy Language)
- meta-tool approach
- tactics
- quickcheck
- programming language
- extensible (Eisbach)
- extensive proof search
- low memory usage
- parallel search
- runtime tactic generation
- efficient proof generation
- native Isabelle proof script
- no code clutter!!

Efficient proof generation with almost no code clutter!!

Efficient proof generation with almost no code clutter!!

Efficient proof generation with almost no code clutter!!

Efficient proof generation with almost no code clutter!!

Efficient proof generation with almost no code clutter!!

Efficient proof generation with almost no code clutter!!

Efficient proof generation with almost no code clutter!!

Efficient proof generation with almost no code clutter!!

Efficient proof generation with almost no code clutter!!
Isabelle/HOL with PSL

proof goal

context

strategy

proof goal

context

tactic / sub-tool

PSL

Much less interaction with Isabelle.
Tactics 1

Case 1
new goal \rightarrow \text{imp} \rightarrow \text{goal}

Case 2
\text{goal} \rightarrow \text{imp} \rightarrow P

Case 3
\text{subgoal 1} \rightarrow \text{imp} \rightarrow \text{subgoal 2} \rightarrow \text{imp} \rightarrow \cdots \rightarrow \text{imp} \rightarrow \text{goal}

principle of explosion
False \rightarrow \text{imp} \rightarrow P

goal \rightarrow \text{preproces} \rightarrow \text{goal} \rightarrow \text{imp} \rightarrow \text{goal}
Tactics 2

Case 1
- new goal
- imp
- goal

Case 2
- goal

Case 3
- subgoal 1
- imp
- subgoal 2
- imp
- ... imp
- goal

: thm
Tactics 2

Case 4 (failure = empty list)
fun tactic :: thm -> [ thm ]

Tactics 3

goal :: thm
tactic
goal 1:: thm
goal 2 :: thm

auto
simp
induct

induct
THEN
auto

simp
OR
auto

REPEAT
simp
Tactical (THEN)

goal :: thm

tactic1

THEN

tactic2

giant tactic?

[...]

@[

[...]

@[

[...]

[...]

@[

[...]

Towards Smart Proof Search. I Yutaka Nagashima
Towards Smart Proof Search. | Yutaka Nagashima

Giant tactic

**problem 1:** Default tactics are too weak!

**problem 2:** Giant tactics are too slow!

**problem 3:** Sledgehammer and quick-check are not tactics!
Towards Smart Proof Search.

Yutaka Nagashima

Thens [**Dynamic**(Induct), Auto, IsSolved]

runtime interpretation

(InductA ++ InductB ++ …) THEN auto THEN is_solved

sequential combination (THEN)

non-determinism
Towards Smart Proof Search. | Yutaka Nagashima

Monadic interpretation

type tactic = thm -> thm Seq.seq

type 'a tactic = 'a -> 'a monad

explicit tree construction?

pointer?

goal

Dynamic (Induct)

Auto

IsSolved

writer monad + non-deterministic monad

efficient proof scripts as “state”
Towards Smart Proof Search.

Yutaka Nagashima

Sledgehammer as tactic

They work on Proof.state not on thm.

problem 3: Sledgehammer and quick-check are not tactics!

\[
\text{type } 'a \text{ tactic } = 'a \rightarrow 'a \text{ nondet_state}
\]

_\text{monad}

\[
\text{type tactic } = \text{P.state } \rightarrow \text{P.state } \text{ nondet_state_monad}
\]

\text{persistant hammering}

\text{Thens [Dyn (Induct), Thens [Hammer+, IsSolved]}

\text{parallel PThenOne Thens [Dyn]}

15 Towards Smart Proof Search. I Y
strategy Basic =
Ors [
    Auto_Solve, 
    Blast_Solve, 
    FF_Solve, 
Thens [IntroClasses, Auto_Solve], 
Thens [Transfer, Auto_Solve], 
Thens [Normalization, IsSolved], 
Thens [DInduct, Auto_Solve], 
Thens [Hammer, IsSolved], 
Thens [DCases, Auto_Solve], 
Thens [DCoinduction, Auto_Solve], 
Thens [Auto, RepeatN(Hammer), IsSolved], 
Thens [DAuto, IsSolved]]

strategy Try_Hard =
Ors [Thens [Subgoal, Basic], 
    Thens [DInductTac, Auto_Solve], 
    Thens [DCaseTac, Auto_Solve], 
    Thens [Subgoal, Advanced], 
    Thens [DCaseTac, Solve_Many], 
    Thens [DInductTac, Solve_Many] ]
PSL: Demo
PSL and try-hard for Isabelle/HOL

The percentage of automatically proved obligations out of 1526 proof obligations (timeout = 300s)

Part 1

try_hard

Part 2

try_smart

sledgehammer

0%

25%

50%

75%

100%

73%

28%

20%

16%

57%
Towards Smart Proof Search.

PaMpeR: Proof Method Recommendation System

huge and complex

proof goal

context

strategy

proof goal

context

assertions

proof goal and context as a vector of boolean values

Type class mechanism? Recursively defined constant?

e.g. AFP & seL4

PaMpeR

Regression Algorithm

Proof Data Base

proof method recommendation::
(proof method * double) list
PaMpeR: Demo

Affine_Arithmetic/Affine_Approximation
Future work: try-hard to try-smart

PaMpeR

proof goal  context
small strategy
proof goal  context
tactic / sub-tool

try_smart

proof goal  context

Efficient tactic

Even better than PSL.

state monad transformer
runtime tactic generation
Towards Smart Proof Search.

Yutaka Nagashima

Isabelle/PSL on Github
(https://github.com/data61/PSL)
Leave a star if you like.

I want you to use PSL / adopt the idea

Lean/PSL coming soon(?)

Isabelle/PaMpeR on Github (still work in progress)
Thank You

TS/ProofEngineering
Yutaka Nagashima
Engineer