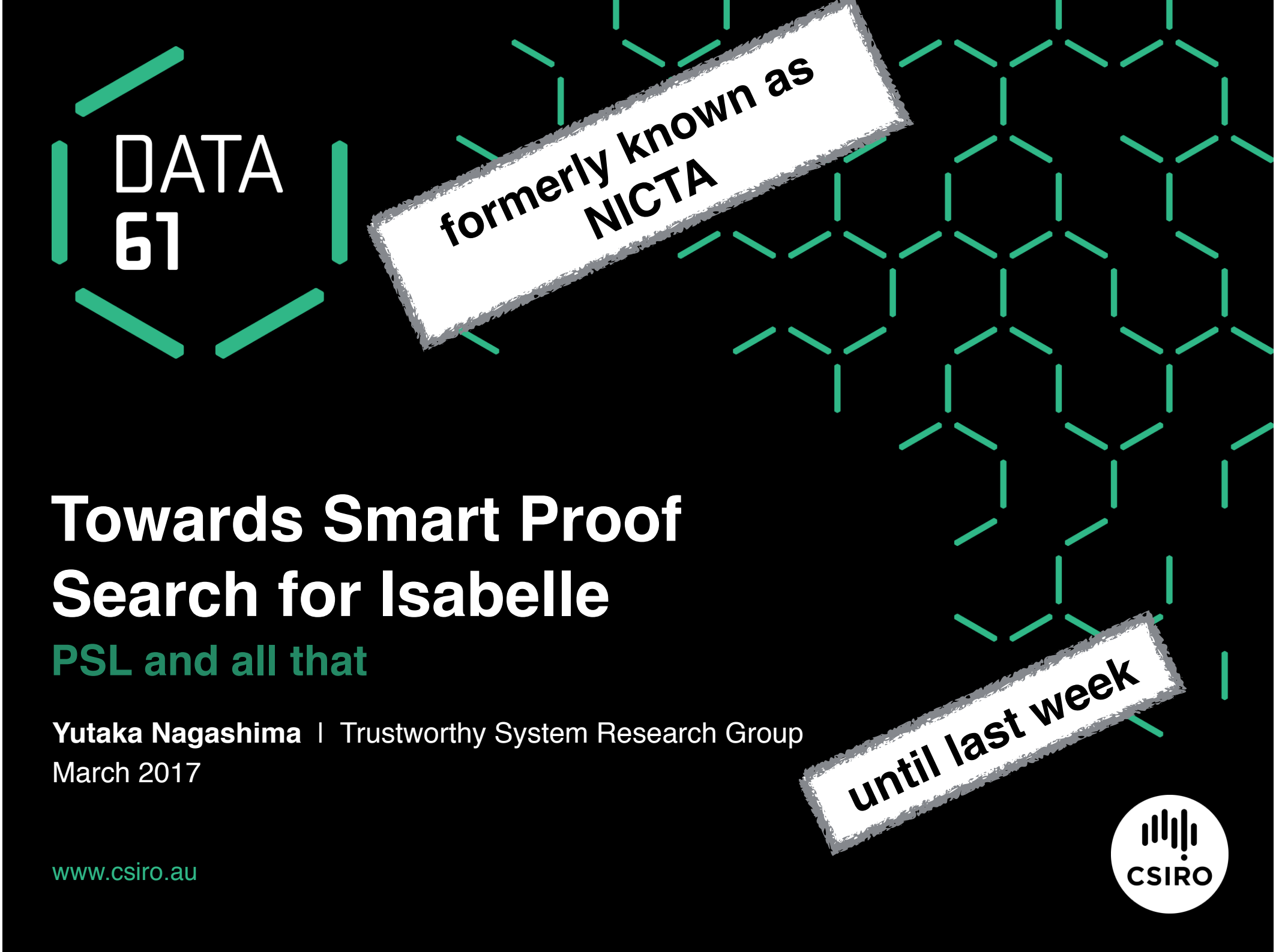

DATA
61
formerly known as
NICTA
Towards Smart Proof
Search for Isabelle
PSL and all that
Yutaka Nagashima | Trustworthy System Research Group
March 2017
until last week
www.csiro.au
CSIRO

Example proof at Data61

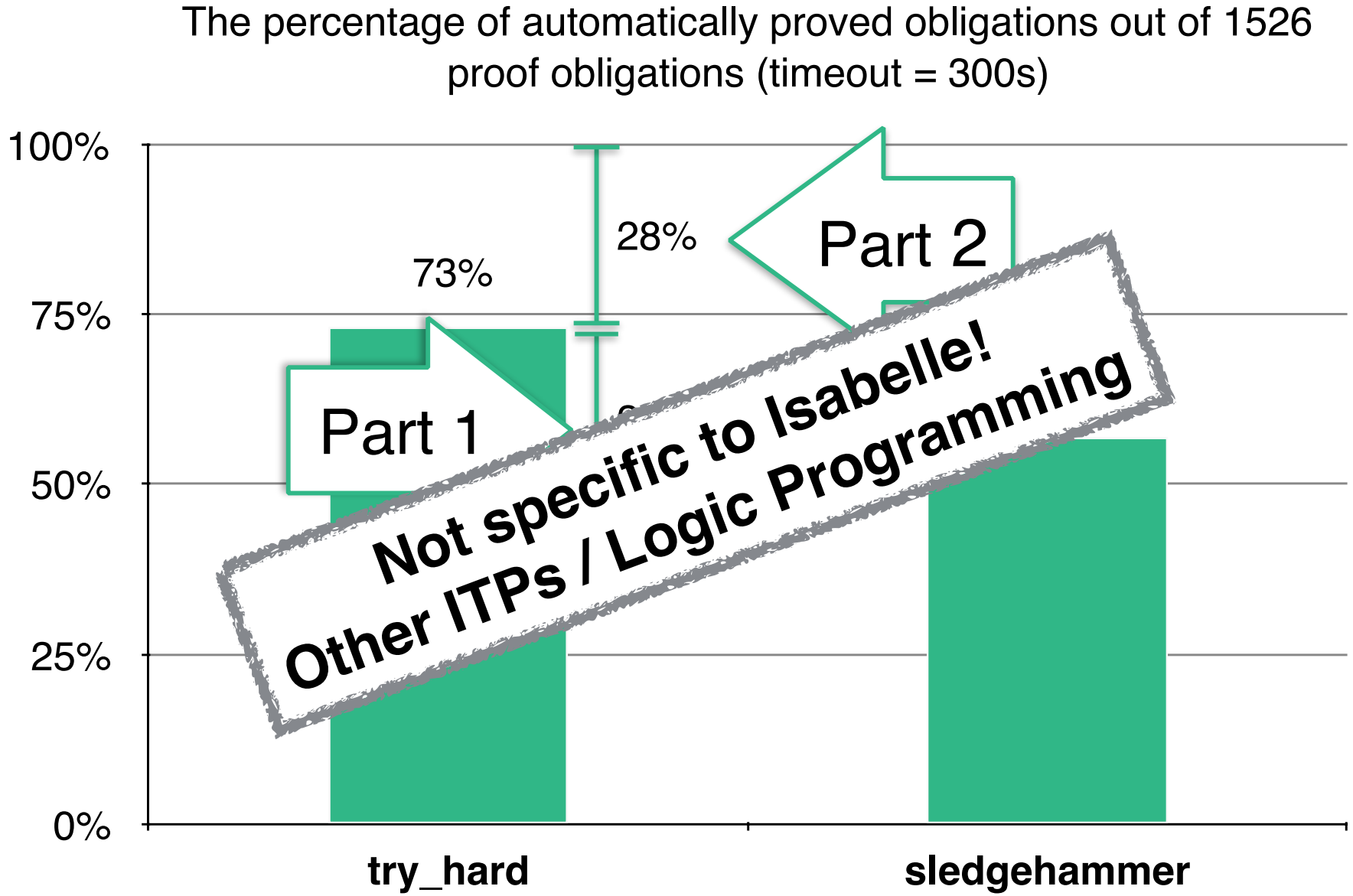
```
39  lemma performPageTableInvocationUnmap_ccorres:
40    "ccorres (K (K \<bottom>) \<currency> dc) (liftxf errstate id (K ()) ret__unsig
41         (invs' and cte_wp_at' (diminished' (ArchObjectCap cap) \<circ> cteCap) ctS
42                and (\<lambda>_. isPageTableCap cap))
43         (UNIV \<inter> \<lbrace>ccap_relation (ArchObjectCap cap) \<acute>cap\<rbra
44         []
45         (liftE (performPageTableInvocation (PageTableUnmap cap ctSlot)))
46         (Call performPageTableInvocationUnmap_'proc)"
47    apply (simp only: liftE_liftM ccorres_liftM_simp)
48    apply (rule ccorres_gen_asm)
49    apply (cinit lift: cap_' ctSlot_')
50     apply csymbr
```

taken from:
https://github.com/seL4/seL4

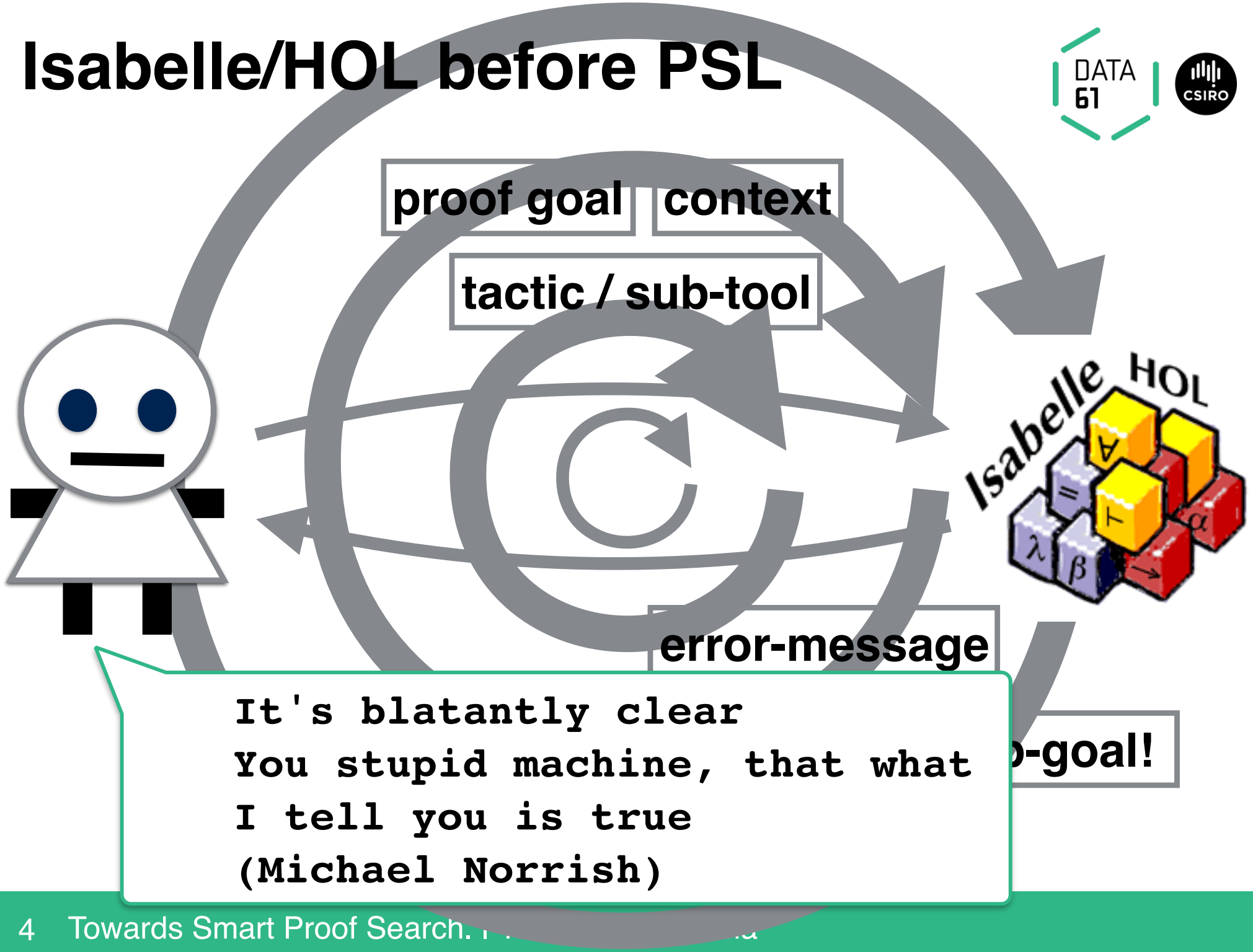
The salary range for this position is AUD 65,000 to 90,000 for recent graduates,

```
53        apply (subgoal_tac "capPTMappedAddress cap
54                              = (\<lambda>cp. if to_bool (capPTIsMapped_CL cp)
55                                 then Some (capPTMappedASID_CL cp, capPTMappedAddres
56                                 else None) (cap_page_table_cap_lift capa)")
57         apply (rule ccorres_Cond_rhs)
58          apply (simp add: to_bool_def)
```

PSL and try-hard for Isabelle/HOL



Isabelle/HOL before PSL



PSL (Proof Strategy Language)

meta-tool approach

tactics

quickcheck

sledgehammer

programming language

extensible (Eisbach)

PSL

runtime tactic generation

efficient proof generation

extensive proof search

parallel search

low memory usage

almost no code clutter!!

ative Isabelle proof script

Isabelle/HOL with PSL

Much less interaction with Isabelle.

Tactics 1

goal → preproces → goal → imp → goal

tactic

principle of explosion

False → imp → P

Case 1

new goal → imp → goal

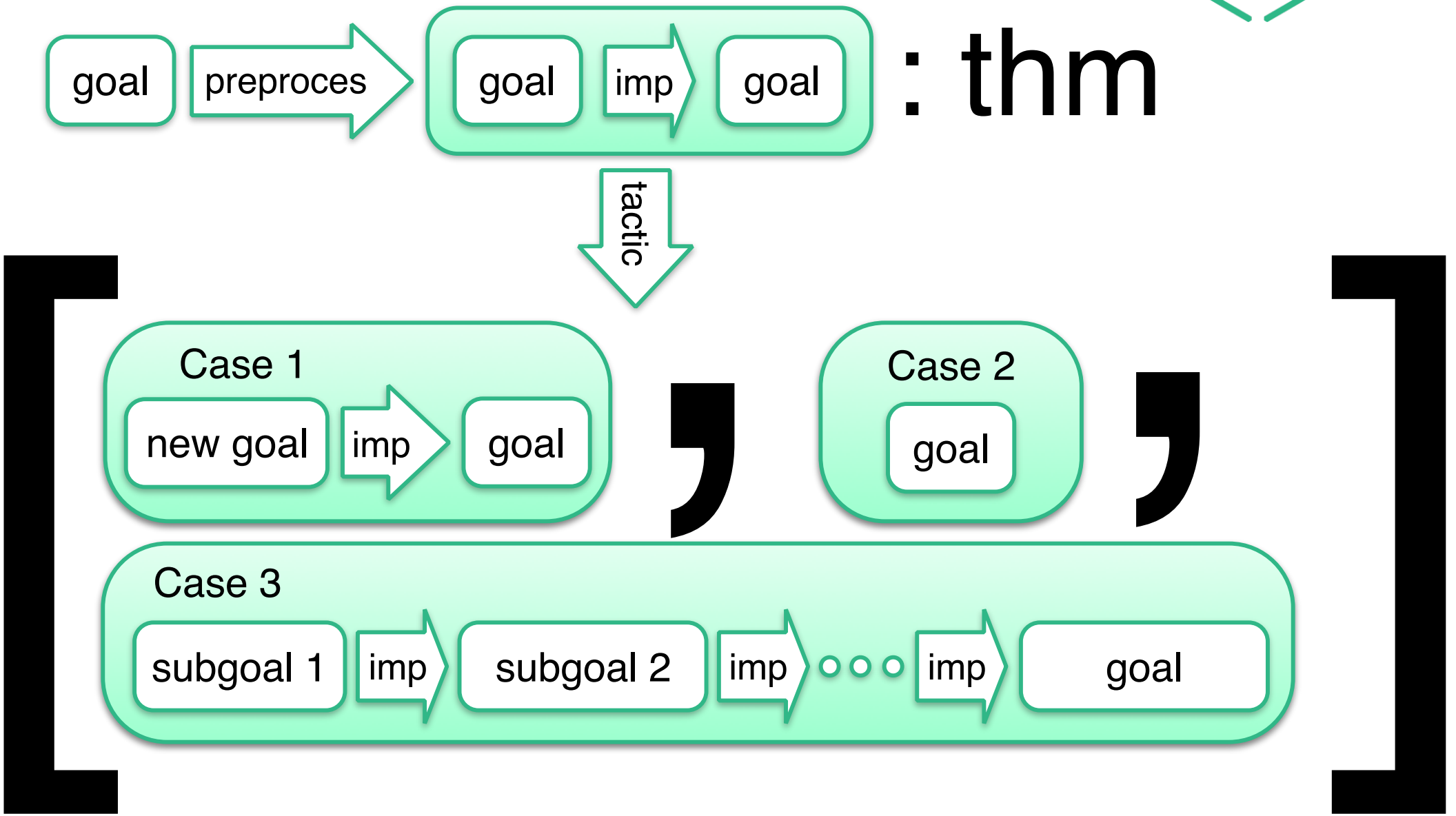
Case 2

goal

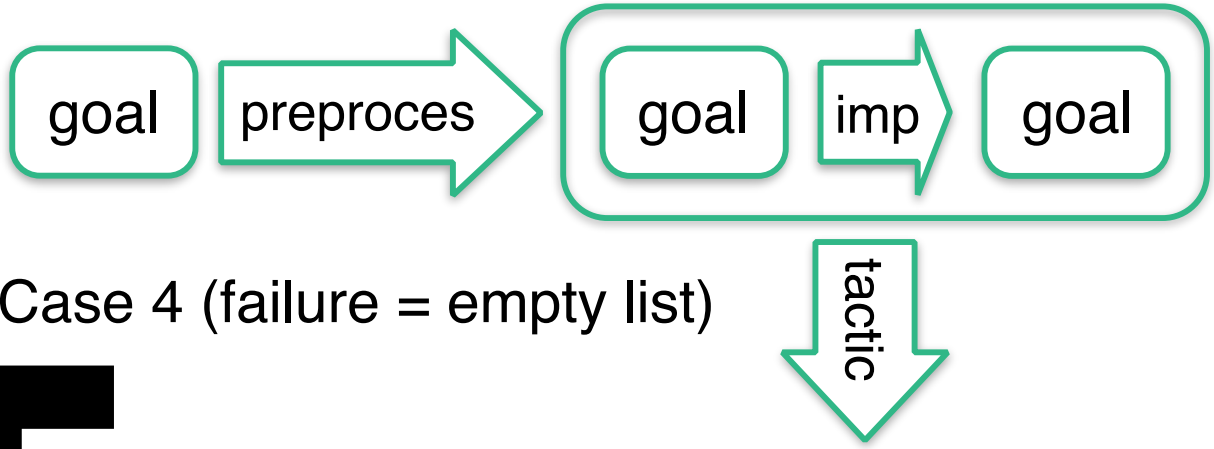
Case 3

subgoal 1 → imp → subgoal 2 → imp → ○○○ → imp → goal

Tactics 2


goal
preproces
goal
imp
goal
: thm
tactic
[
Case 1
new goal
imp
goal
,
Case 2
goal
,
Case 3
subgoal 1
imp
subgoal 2
imp
imp
goal
]

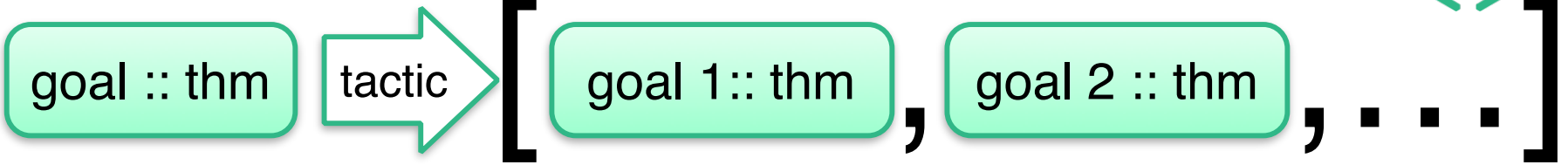
Tactics 2



Case 4 (failure = empty list)

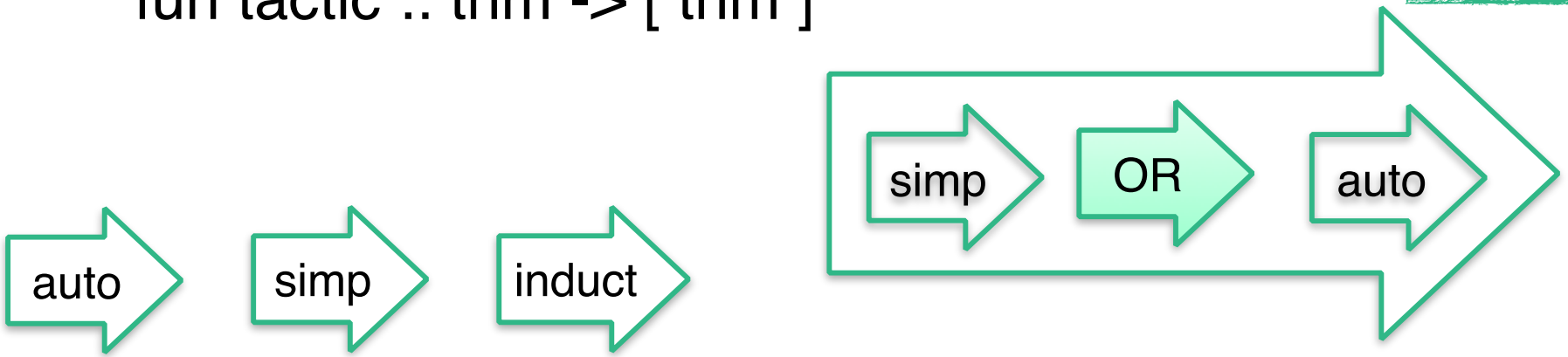
[ ]

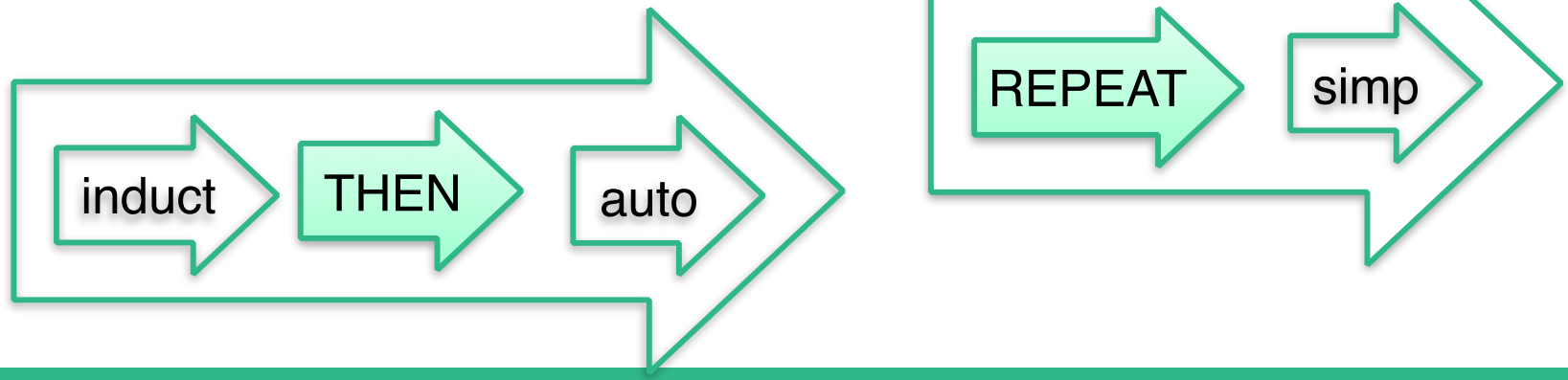
Tactics 3



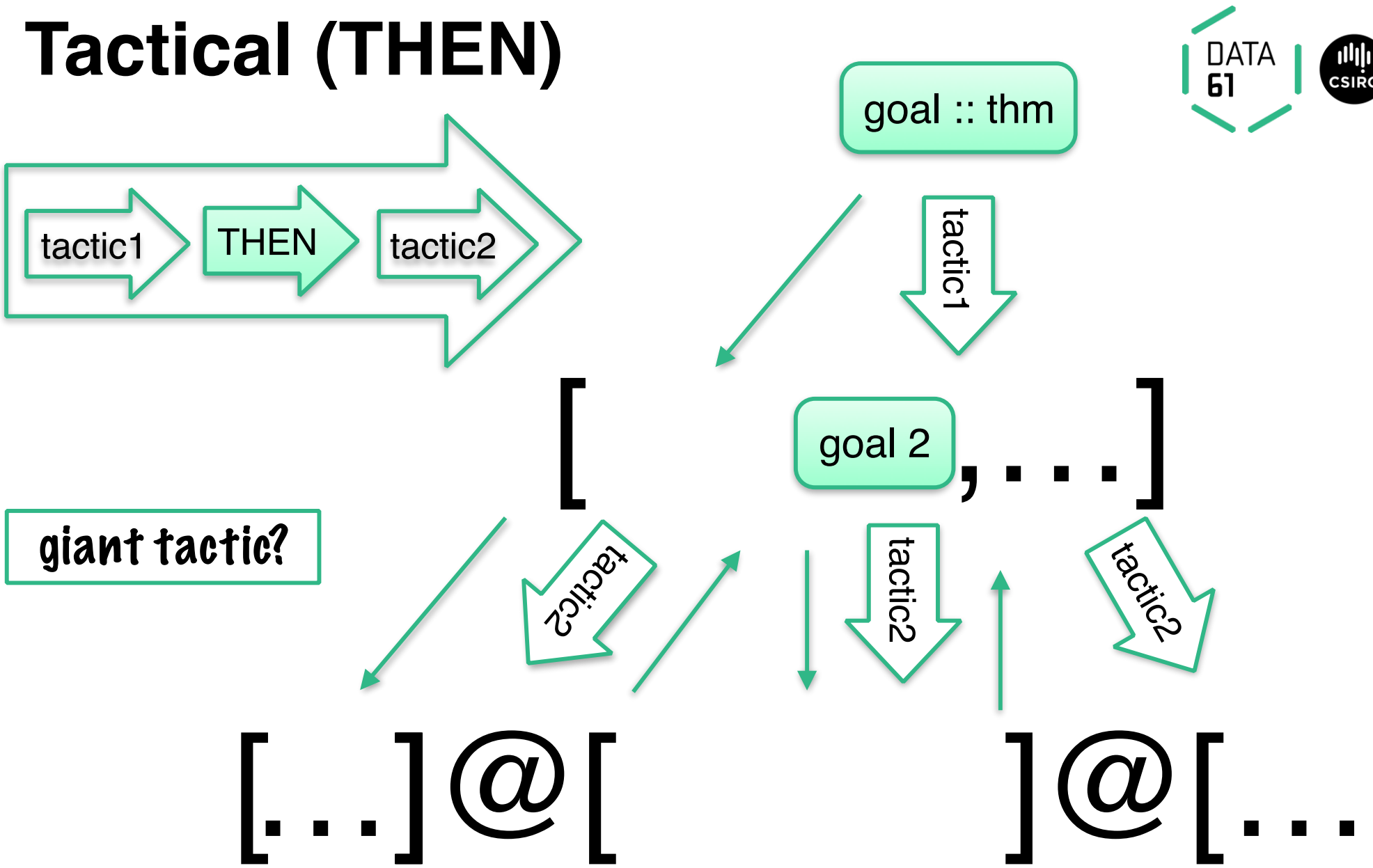
Lazy

fun tactic :: thm -> [thm]



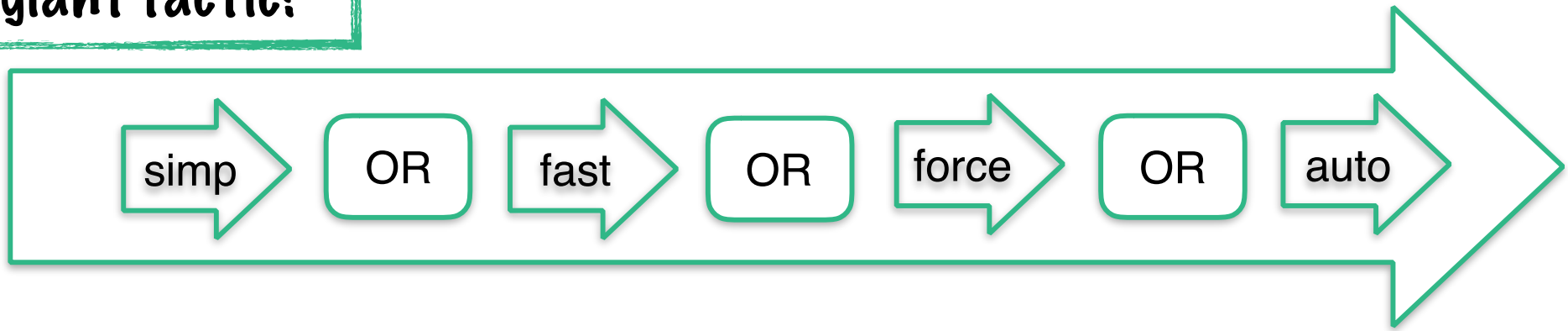


Tactical (THEN)



Giant tactic

giant tactic?



problem 1: Default tactics are too weak!

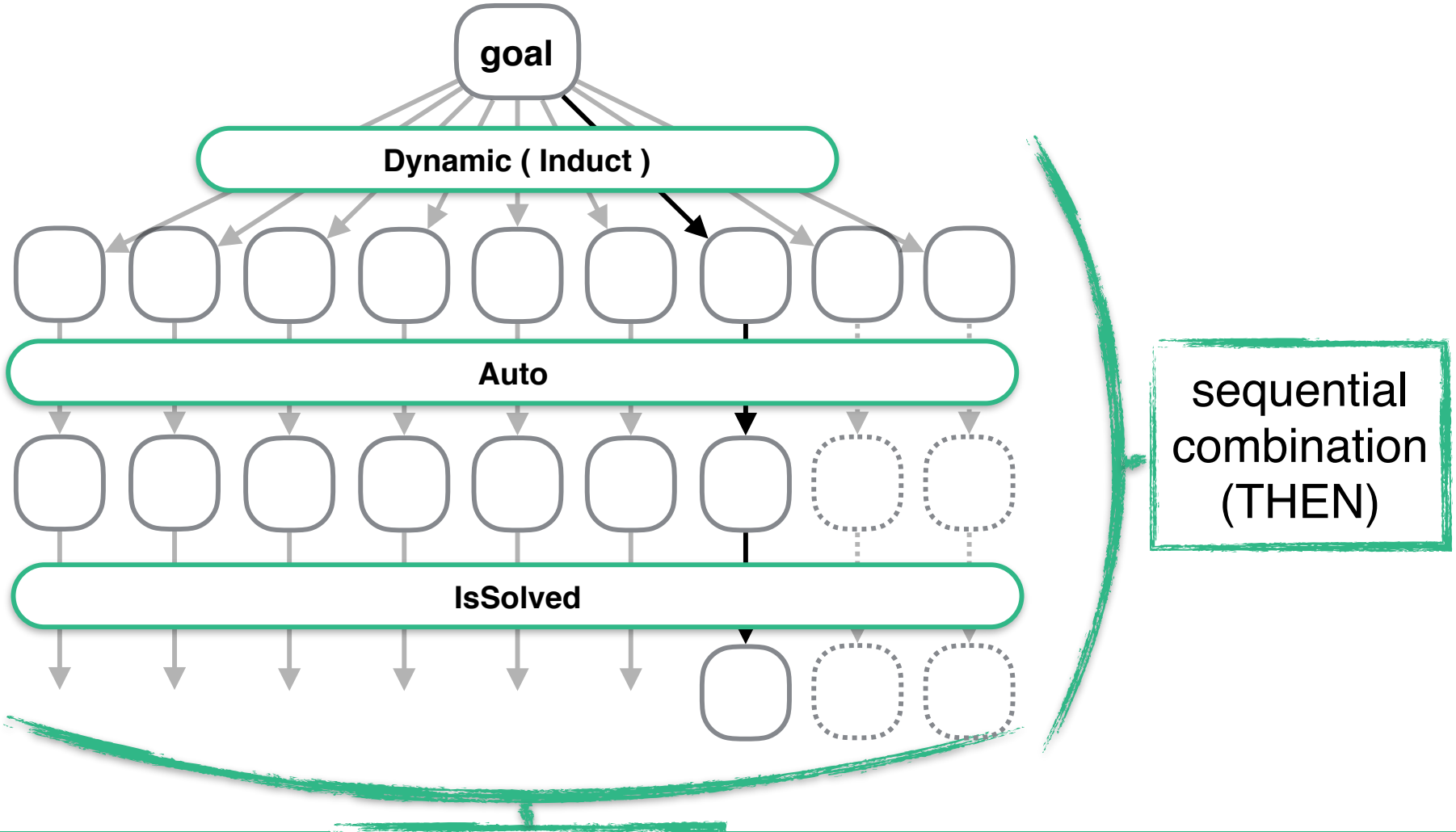
problem 2: Giant tactics are too slow!

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

Thens [Dynamic(Induct), Auto, IsSolved]

runtime interpretation

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



Monadic interpretation

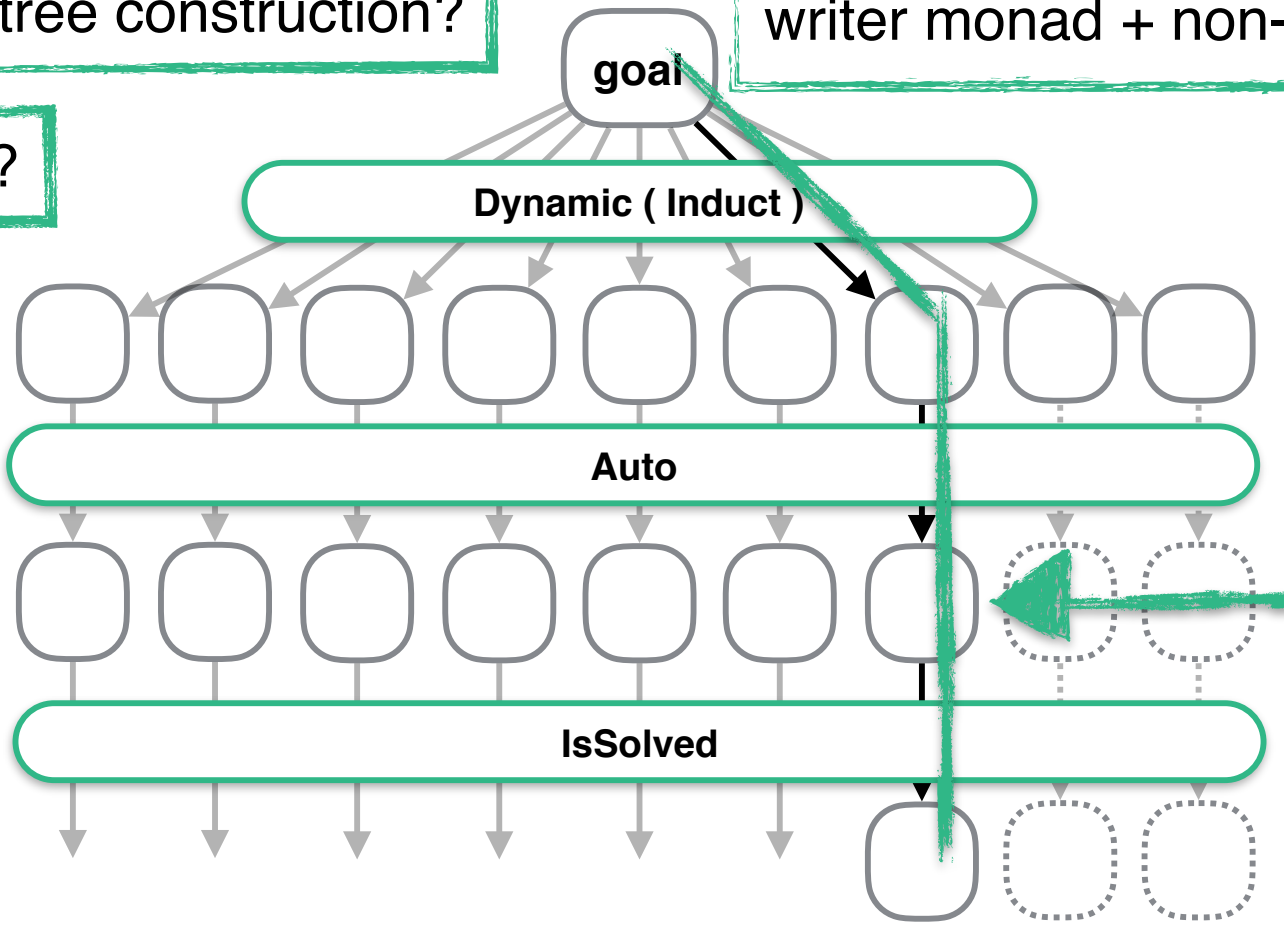
type tactic = thm -> thm Seq.seq

type ‘a tactic = ‘a -> ‘a monad

explicit tree construction?

writer monad + non-deterministic monad

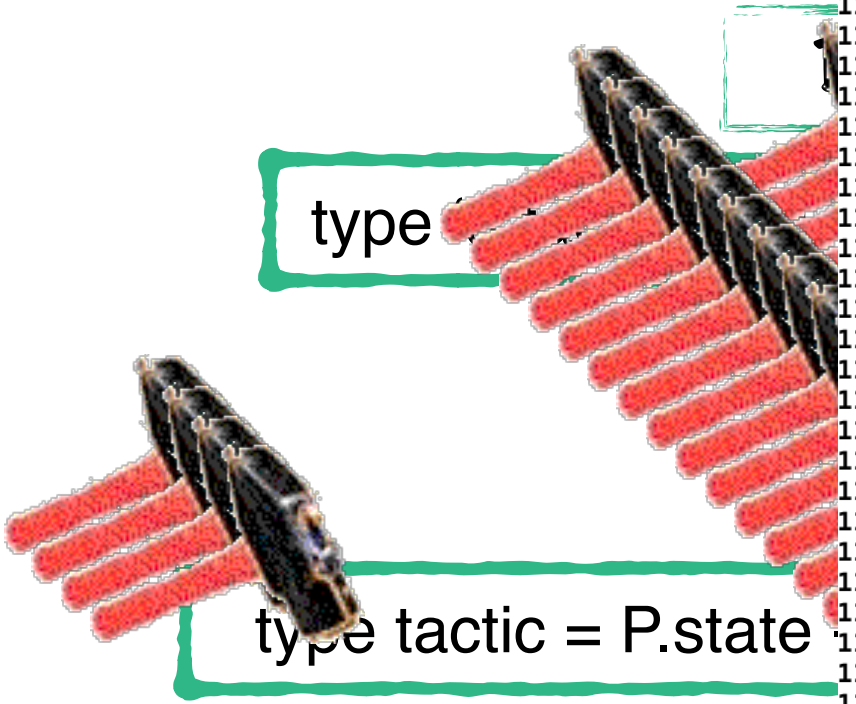
pointer?



efficient proof scripts as “state”

Sledgehammer a

problem 3: Sledgehammer al



type

type tactic = P.state

parallel

persistant hamm

PThenOne Th ns [Dyn

Untitled | Untitled

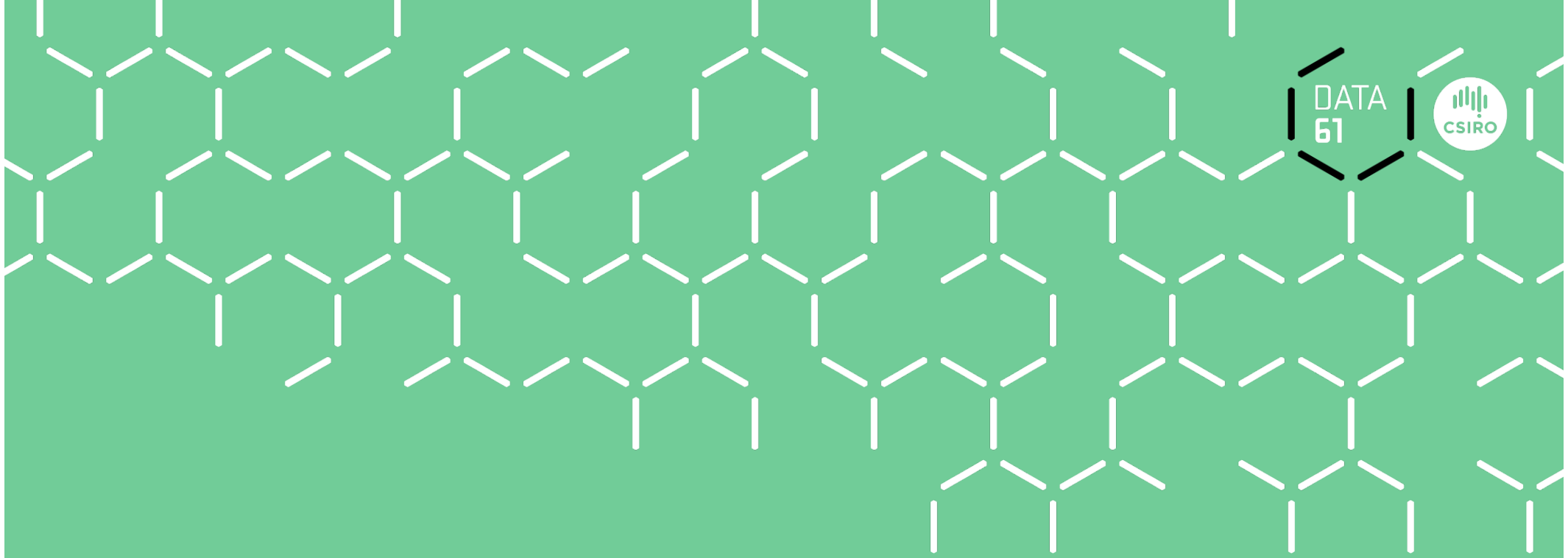
```
Tasks: 712 total,  48 running, 664 sleeping,   0 stopped,   0 zombie
%Cpu(s): 94.8 us,  2.2 sy,  0.0 ni,  3.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 s
KiB Mem : 26397948+total, 25078707+free,  9756756 used,  3435664 buff/cache
KiB Swap: 11891708 total, 11891708 free,        0 used. 25261203+avail Mem

   PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAN
110381 yutaka    20   0 3384924 1.685g   7944 S 585.6   0.7  47:26.3 poly
119078 yutaka    20   0  141192 118764  10996 R 100.0   0.0   0:06.8 cvc4
119018 yutaka    20   0  128416 106196  10996 R 100.0   0.0   0:07.2 cvc4
119030 yutaka    20   0   86556  64956  11060 R 100.0   0.0   0:07.8 cvc4
119042 yutaka    20   0   90732  69256  11060 R 100.0   0.0   0:06.8 cvc4
119052 yutaka    20   0  118240  96036  10996 R 100.0   0.0   0:06.9 cvc4
119085 yutaka    20   0  128412 106168  10996 R 100.0   0.0   0:06.1 cvc4
119102 yutaka    20   0   83348  62116  11124 R 100.0   0.0   0:06.8 cvc4
119106 yutaka    20   0   83880  62844  11060 R 100.0   0.0   0:06.7 cvc4
119110 yutaka    20   0  128416 105936  10996 R 100.0   0.0   0:06.8 cvc4
119118 yutaka    20   0  119556  98244  10996 R 100.0   0.0   0:06.0 cvc4
119126 yutaka    20   0  117928  96176  10996 R 100.0   0.0   0:05.4 cvc4
119138 yutaka    20   0  117916  96396  10996 R 100.0   0.0   0:05.9 cvc4
119154 yutaka    20   0   82164  61052  11124 R 100.0   0.0   0:05.9 cvc4
119174 yutaka    20   0  117944  96432  10996 R 100.0   0.0   0:05.6 cvc4
119192 yutaka    20   0   72612  51720  10932 R 100.0   0.0   0:05.2 cvc4
119198 yutaka    20   0  125328 103624  10996 R 100.0   0.0   0:05.0 cvc4
119210 yutaka    20   0   80492  59224  11124 R 100.0   0.0   0:05.4 cvc4
119218 yutaka    20   0   73820  53296  10996 R 100.0   0.0   0:05.0 cvc4
119250 yutaka    20   0  154872 132780  10996 R 100.0   0.1   0:05.7 cvc4
119262 yutaka    20   0  103472  81892  10996 R 100.0   0.0   0:05.4 cvc4
119266 yutaka    20   0   72348  51460  10932 R 100.0   0.0   0:05.2 cvc4
118954 yutaka    20   0  139324 115908  11060 R 100.0   0.0   0:09.0 cvc4
118994 yutaka    20   0   84740  63188  11124 R 100.0   0.0   0:08.9 cvc4
119006 yutaka    20   0  175804 153276  10996 R 100.0   0.1   0:07.3 cvc4
119066 yutaka    20   0   85660  64168  11060 R 100.0   0.0   0:06.3 cvc4
119086 yutaka    20   0  128412 106180  10996 R 100.0   0.0   0:06.1 cvc4
119114 yutaka    20   0  125620 103496  10996 R 100.0   0.0   0:06.7 cvc4
119150 yutaka    20   0  117928  96408  10996 R 100.0   0.0   0:05.8 cvc4
119182 yutaka    20   0   82968  61544  11060 R 100.0   0.0   0:05.3 cvc4
119202 yutaka    20   0   82964  61788  11060 R 100.0   0.0   0:05.5 cvc4
119222 yutaka    20   0  123400 101416  10996 R 100.0   0.0   0:05.9 cvc4
119226 yutaka    20   0   97524  75872  10996 R 100.0   0.0   0:05.0 cvc4
119234 yutaka    20   0   80480  59176  11060 R 100.0   0.0   0:05.5 cvc4
118970 yutaka    20   0  128416 106200  10996 R 100.0   0.0   0:08.9 cvc4
119130 yutaka    20   0  159592 136772  10996 R 100.0   0.1   0:05.7 cvc4
119160 yutaka    20   0   83216  62120  11124 R 100.0   0.0   0:05.8 cvc4
119170 yutaka    20   0  117916  96396  10996 R 100.0   0.0   0:05.4 cvc4
119254 yutaka    20   0  168652 145240  10996 R 100.0   0.1   0:05.1 cvc4
118946 yutaka    20   0  128412 106168  10996 R 100.0   0.0   0:09.1 cvc4
118974 yutaka    20   0  128412 106188  10996 R 100.0   0.0   0:08.7 cvc4
118986 yutaka    20   0   84760  63200  11124 R 100.0   0.0   0:08.8 cvc4
119060 yutaka    20   0  128416 106132  10996 R 100.0   0.0   0:06.9 cvc4
119194 yutaka    20   0  115752  94176  10996 R 100.0   0.0   0:05.9 cvc4
118966 yutaka    20   0  128416 106148  10996 R  99.7   0.0   0:08.8 cvc4
```

try_hard: the default strategy

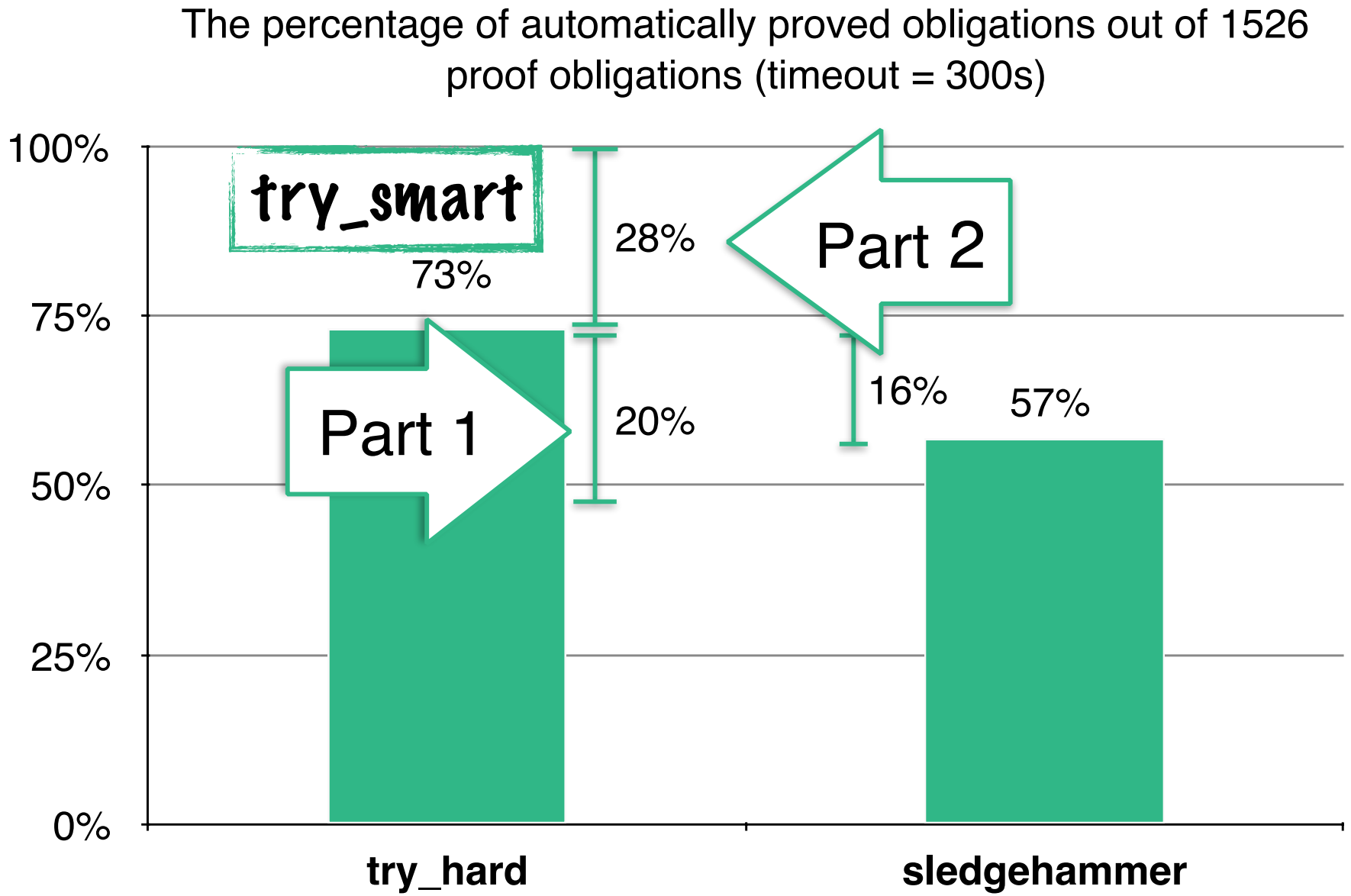
```
strategy Try_Hard =
Ors [Thens [Subgoal, Basic],
      Thens [DInductTac, Auto_Solve],
      Thens [DCaseTac, Auto_Solve],
      Thens [Subgoal, Advanced],
      Thens [DCaseTac, Solve_Many],
      Thens [DInductTac, Solve_Many] ]
```

```
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]]
```



PSL:
Demo

PSL and try-hard for Isabelle/HOL



PaMpeR: Proof M

huge and complex

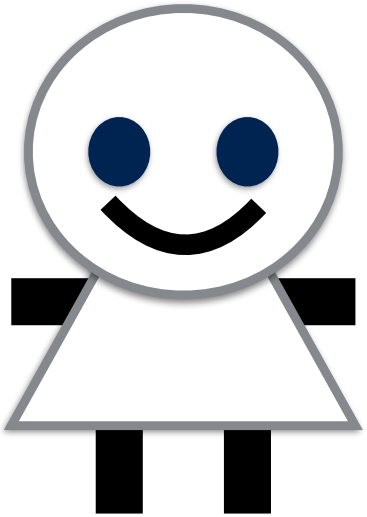
proof goal

context | strategy

proof goal and context as a vector of boolean values

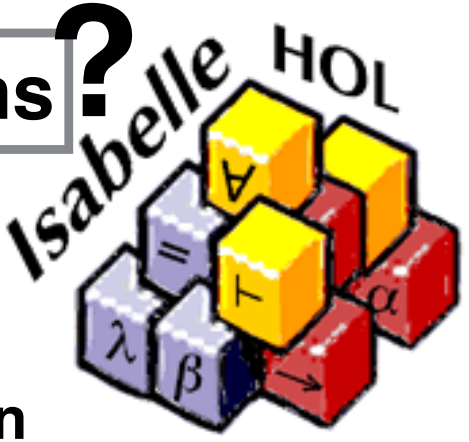
proof goal

context | assertions ?



PaMpeR

Regression Algorithm



Proof Data Base

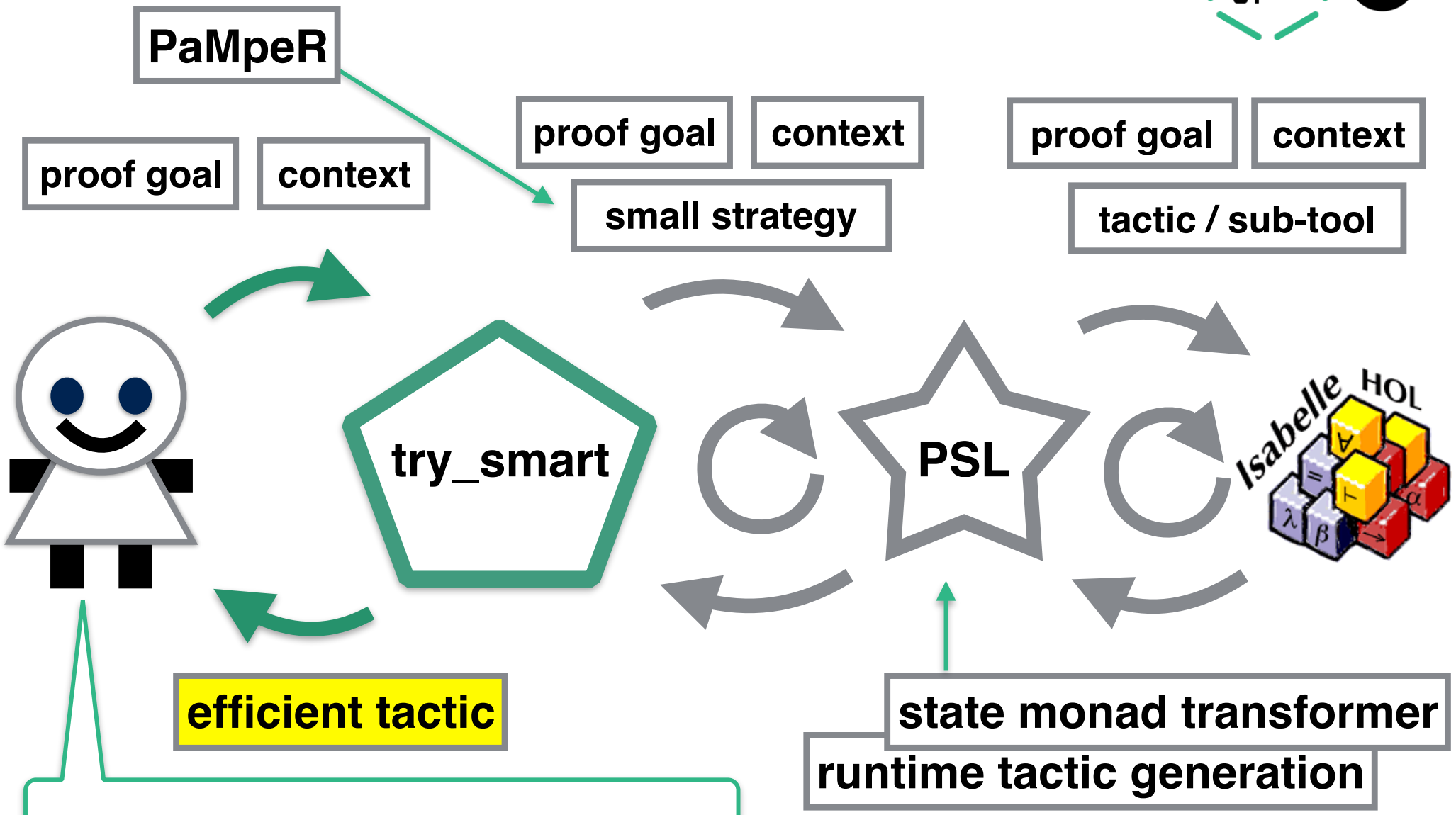
Type class mechanism?
Recursively defined constant?
e.g. AFP & seL4

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

PaMpeR:
Demo

Affine_Arithmetic/Affine_Approximation

Future work: try-hard to try-smart



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

www.csiro.au