Cooperating Proof Attempts in Vampire

Giles Reger Dmitry Tishkovsky Andrei Voronkov

University of Manchester

6th August 2015

Outline

Simple Idea

- Very simple idea:

  Run more than one proof attempt, have them cooperate
- Lots of previous work
  - Strategy selection in Gandelf with clause reuse
  - Parallel proving with clause sharing in DISCOUNT
  - ...
- But these lacked a good vehicle for cooperation
- This work is about cooperation between concurrently running proof attempts

  ...but supporting parallelism is a goal
- We didn't use these ideas in this year's CASC competition
- Firstly, why multiple proof attempts?

Vampire Options

1. age weight ratio
2. backward demodulation
3. binary resolution
4. backward subsumption
5. backward subsumption resolution
6. congruence closure unsat cores
7. condensation
8. dismatching constraints
9. equality proxy
10. extensionality resolution
11. function definition elimination
12. fmb symmetry ratio
13. forward subsumption resolution
14. global subsumption (gs)
15. gs avatar assumptions
16. gs explicit minimisation
17. gs sat solver power
18. general splitting
19. instgen big restart ratio
20. instgen passive reactivation
21. instgen restart period quotient
22. instgen resolution ratio
23. instgen selection
24. instgen with resolution
25. inequality splitting
26. instantiation
27. increased numeral weight
28. literal comparison mode
29. lrs weight limit only
30. nonliterals in clause weight
31. naming
32. nongoal weight coefficient
33. saturation algorithm
34. selection
35. splitting (spl)
36. spl add complementary
37. spl delete deactivated
38. spl fast restart
39. spl minimise model
40. spl add complementary
41. spl with congruence closure
42. spl eager removal
43. spl flushing period
44. spl flushing quotient
45. spl non-splittable components
46. sat solver
47. sine selection
48. sine depth
49. sine tolerance
50. symbol precedence
51. set of support
52. simulated time limit
53. time limit
54. theory axioms
55. theory flattening
56. unused predicate removal
57. unit resulting resolution

Vampire Options

1. age weight ratio
2. backward demodulation
3. binary resolution
4. backward subsumption
5. backward subsumption resolution
6. congruence closure unsat cores
7. condensation
8. dismatching constraints
9. equality proxy
10. extensionality resolution
11. function definition elimination
12. fmb symmetry ratio
13. forward subsumption resolution
14. global subsumption (gs)
15. gs avatar assumptions
16. gs explicit minimisation
17. gs sat solver power
18. general splitting
19. instgen big restart ratio
20. instgen passive reactivation
21. instgen restart period quotient
22. instgen resolution ratio
23. instgen selection
24. instgen with resolution
25. inequality splitting
26. instantiation
27. increased numeral weight
28. literal comparison mode
29. lrs weight limit only
30. nonliterals in clause weight
31. naming
32. nongoal weight coefficient
33. saturation algorithm
34. selection
35. splitting (spl)
36. spl add complementary
37. spl delete deactivated
38. spl fast restart
39. spl minimise model
40. spl add complementary
41. spl with congruence closure
42. spl eager removal
43. spl flushing period
44. spl flushing quotient
45. spl non-splittable components
46. sat solver
47. sine selection
48. sine depth
49. sine tolerance
50. symbol precedence
51. set of support
52. simulated time limit
53. time limit
54. theory axioms
55. theory flattening
56. unused predicate removal
57. unit resulting resolution

Vampire Options

1. age weight ratio
2. backward demodulation
3. binary resolution
4. backward subsumption
5. backward subsumption resolution
6. congruence closure unsat cores
7. condensation
8. dismatching constraints
9. equality proxy
10. extensionality resolution
11. function definition elimination
12. fmb symmetry ratio
13. forward subsumption resolution
14. global subsumption (gs)
15. gs avatar assumptions
16. gs explicit minimisation
17. gs sat solver power
18. general splitting
19. instgen big restart ratio
20. instgen passive reactivation
21. instgen restart period quotient
22. instgen resolution ratio
23. instgen selection
24. instgen with resolution
25. inequality splitting
26. instantiation
27. increased numeral weight
28. literal comparison mode
29. lrs weight limit only
30. nonliterals in clause weight
31. naming
32. nongoal weight coefficient
33. saturation algorithm
34. selection
35. splitting (spl)
36. spl add complementary
37. spl delete deactivated
38. spl fast restart
39. spl minimise model
40. spl add complementary
41. spl with congruence closure
42. spl eager removal
43. spl flushing period
44. spl flushing quotient
45. spl non-splittable components
46. sat solver
47. sine selection
48. sine depth
49. sine tolerance
50. symbol precedence
51. set of support
52. simulated time limit
53. time limit
54. theory axioms
55. theory flattening
56. unused predicate removal
57. unit resulting resolution

Vampire Options

1. age weight ratio
2. backward demodulation
3. binary resolution
4. backward subsumption
5. backward subsumption resolution
6. congruence closure unsat cores
7. condensation
8. dismatching constraints
9. equality proxy
10. extensionality resolution
11. function definition elimination
12. fmb symmetry ratio
13. forward subsumption resolution
14. global subsumption (gs)
15. gs avatar assumptions
16. gs explicit minimisation
17. gs sat solver power
18. general splitting
19. instgen big restart ratio
20. instgen passive reactivation
21. instgen restart period quotient
22. instgen resolution ratio
23. instgen selection
24. instgen with resolution
25. inequality splitting
26. instantiation
27. increased numeral weight
28. literal comparison mode
29. lrs weight limit only
30. nonliterals in clause weight
31. naming
32. nongoal weight coefficient
33. saturation algorithm
34. selection
35. splitting (spl)
36. spl add complementary
37. spl delete deactivated
38. spl fast restart
39. spl minimise model
40. spl add complementary
41. spl with congruence closure
42. spl eager removal
43. spl flushing period
44. spl flushing quotient
45. spl non-splittable components
46. sat solver
47. sine selection
48. sine depth
49. sine tolerance
50. symbol precedence
51. set of support
52. simulated time limit
53. time limit
54. theory axioms
55. theory flattening
56. unused predicate removal
57. unit resulting resolution

Vampire Options

1. age weight ratio
2. backward demodulation
3. binary resolution
4. backward subsumption
5. backward subsumption resolution
6. congruence closure unsat cores
7. condensation
8. dismatching constraints
9. equality proxy
10. extensionality resolution
11. function definition elimination
12. fmb symmetry ratio
13. forward subsumption resolution
14. global subsumption (gs)
15. gs avatar assumptions
16. gs explicit minimisation
17. gs sat solver power
18. general splitting
19. instgen big restart ratio
20. instgen passive reactivation
21. instgen restart period quotient
22. instgen resolution ratio
23. instgen selection
24. instgen with resolution
25. inequality splitting
26. instantiation
27. increased numeral weight
28. literal comparison mode
29. lrs weight limit only
30. nonliterals in clause weight
31. naming
32. nongoal weight coefficient
33. saturation algorithm
34. selection
35. splitting (spl)
36. spl add complementary
37. spl delete deactivated
38. spl fast restart
39. spl minimise model
40. spl add complementary
41. spl with congruence closure
42. spl eager removal
43. spl flushing period
44. spl flushing quotient
45. spl non-splittable components
46. sat solver
47. sine selection
48. sine depth
49. sine tolerance
50. symbol precedence
51. set of support
52. simulated time limit
53. time limit
54. theory axioms
55. theory flattening
56. unused predicate removal
57. unit resulting resolution

Vampire Strategies

- In CASC 2015 we tried 351 unique strategies
- What do they use?
  - 303 use saturation (128 dis, 128 lrs, 57 ott), 32 instgen, 6 fmb
  - 231 use AVATAR
  - On average vary 13 options, the longest varies 25
  - Time limits: shortest 0.1s, longest 600s, mean 16.1 with sdev 42.4, median 4.3
- What do they solve?
  - 933 solutions, 372 use 1 strategy (561 use more)
  - Mean 3.9 with sdev 5.6, median 2, max 53
  - 152 unique strats (prove mean 6.1 sdev 13, median 2, max 91)

- Observations
  - Very short strategies are useful
  - Lots of complementary strategies are required

Vampire Strategies

- In CASC 2015 we found solutions with 152 unique strategies
- What do they use?
  - 133 use saturation (61 dis, 44 lrs, 28 ott), 13 instgen, 6 fmb
  - 105 use AVATAR
  - On average vary 12 options, the longest varies 25
  - Time limits: shortest 0.1s, longest 600s, mean 26.4 with sdev 61.4, median 5.6
- What do they solve?
  - 933 solutions, 372 use 1 strategy (561 use more)
  - Mean 3.9 with sdev 5.6, median 2, max 53
  - 152 unique strats (prove mean 6.1 sdev 13, median 2, max 91)

- Observations
  - Very short strategies are useful
  - Lots of complementary strategies are required

Vampire Strategies

- In CASC 2015 we found solutions with 152 unique strategies
- What do they use?
  - 133 use saturation (61 dis, 44 lrs, 28 ott), 13 instgen, 6 fmb
  - 105 use AVATAR
  - On average vary 12 options, the longest varies 25
  - Time limits: shortest 0.1s, longest 600s, mean 26.4 with sdev 61.4, median 5.6
- What do they solve?
  - 933 solutions, 372 use 1 strategy (561 use more)
  - Mean 3.9 with sdev 5.6, median 2, max 53
  - 152 unique strats (prove mean 6.1 sdev 13, median 2, max 91)

```
fmb+10_1_sas=minisat_2046
```

- Observations
  - Very short strategies are useful
  - Lots of complementary strategies are required

Vampire Strategies

- In CASC 2015 we found solutions with 152 unique strategies
- What do they use?
  - 133 use saturation (61 dis, 44 lrs, 28 ott), 13 instgen, 6 fmb
  - 105 use AVATAR
  - On average vary 12 options, the longest varies 25
  - Time limits: shortest 0.1s, longest 600s, mean 26.4 with sdev 61.4, median 5.6
- What do they solve?
  - 933 solutions, 372 use 1 strategy (561 use more)
  - Mean 3.9 with sdev 5.6, median 2, max 53
  - 152 unique strats (prove mean 6.1 sdev 13, median 2, max 84)

```
dis-1_4_bd=preordered:cond=fast:fde=none:gs=on:gsssp=full:nwc=1:sas=minisat:sac=on:
sdd=large:sser=off:ssfp=100000:ssfq=1.2:ssnc=none:sp=reverse_arity:updr=off_46
```

- Observations
  - Very short strategies are useful
  - Lots of complementary strategies are required

Vampire Strategies

- In CASC 2015 we found solutions with 152 unique strategies
- What do they use?
  - 133 use saturation (61 dis, 44 lrs, 28 ott), 13 instgen, 6 fmb
  - 105 use AVATAR
  - On average vary 12 options, the longest varies 25
  - Time limits: shortest 0.1s, longest 600s, mean 26.4 with sdev 61.4, median 5.6
- What do they solve?
  - 933 solutions, 372 use 1 strategy (561 use more)
  - Mean 3.9 with sdev 5.6, median 2, max 53
  - 152 unique strats (prove mean 6.1 sdev 13, median 2, max 66)

```
dis+1011_40_bs=on:cond=on:gs=on:gsaa=from_current:nwc=1:sfr=on:ssfp=1000:
ssfq=2.0:smm=sco:ssnc=none:updr=off_282
```

- Observations
  - Very short strategies are useful
  - Lots of complementary strategies are required

This talk

- This works focuses on organising the cooperation of multiple Vampire proof attempts employing different strategies

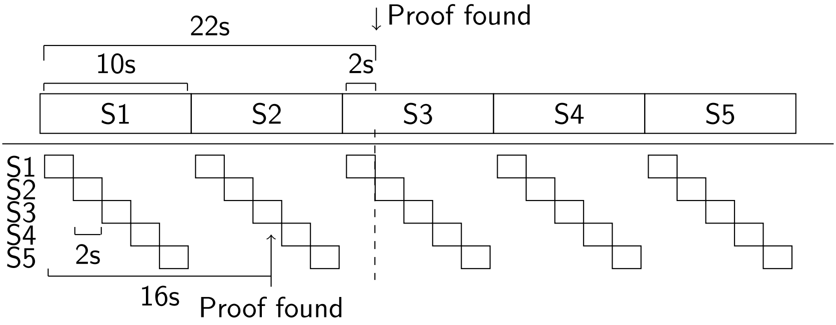
- In this setting we consider two techniques for 'cooperation'
  1. Interleaving of proof attempts to find the short proofs from a single strategy faster
  2. Sharing splitting decisions to prevent a proof attempt from exploring parts of the search space shown not to contain a proof by another proof attempt

Running multiple Proof Attempts...

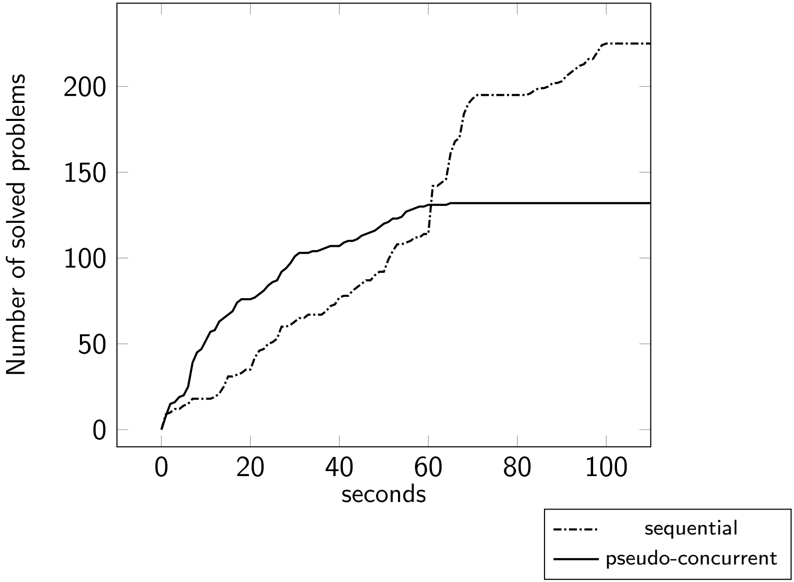
- ... at the same time required us to rewrite quite a bit of Vampire... and introduce an input format for specifying multiple strategies
- Long-term plans to allow proof attempts to run in parallel but currently their execution is interleaved

Interleaving Strategies

- Generally if a strategy finds a proof it finds it quickly
- By interleaving strategies we can find the quick proofs faster



Experiment with just Interleaving



Scheduling

- Lots of variables to play with - still an area of experimentation
- An obvious variable is granularity of interleaving
  - Too small and we get bad memory issues
  - Too big and we don't get the benefit we want
- Other ideas
  - Changing priorities
  - Resource limiting
  - Online learning of 'good' kinds of proof attempts
  - Offline identification of complementary strategies

Proof Search by Saturation

- Vampire is a saturation based prover
- Saturate (up to redundancy) an input set of clauses $\mathcal{C}$ with respect to a set of inferences $\mathcal{I}$
- Pragmatically this involves a growing search space from which clauses are selected and have inferences applied to generate new clauses
- If we derive false then $\mathcal{C}$ was unsatisfiable.
- If we saturate (and $\mathcal{I}$ was complete) then $\mathcal{C}$ was satisfiable

Splitting

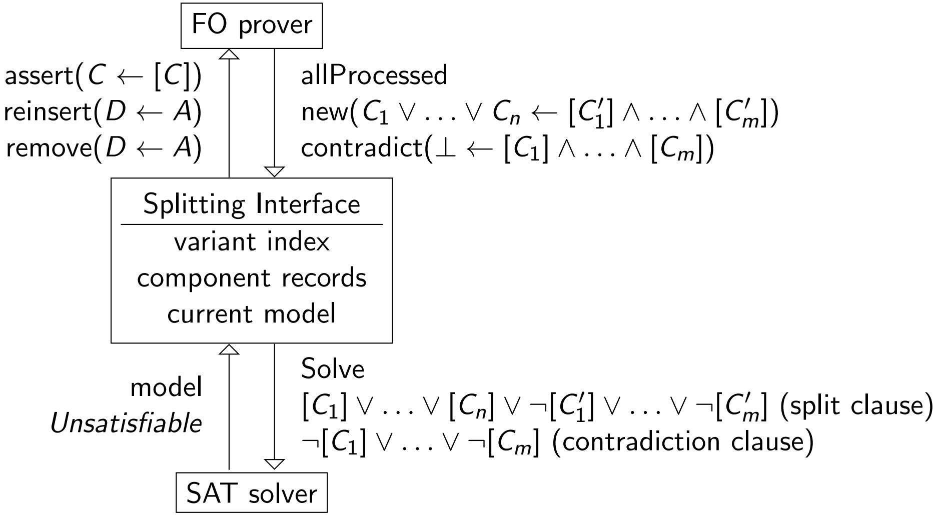
- The search space can become full of long and heavy clauses
- A solution is splitting
    - For variable disjoint clauses C_1 and C_2
    - $S \cup (C_1 \vee C_2)$ is unsat iff both $S \cup C_1$ and $S \cup C_2$ are
    - Consider $S \cup C_1$ and $S \cup C_2$ separately

- For each clause we assert each non-splittable component in turn until all have been refuted or one branch is saturated without refutation

The AVATAR Approach

- The idea: represent the splitting decisions as a SAT problem
- To do this
  1. Name each clause component with a SAT variable
  2. Pass the corresponding SAT clause to a SAT solver
  3. Ask for a model and use this to make splitting decisions
  4. Carry around these assumptions in the first-order part
  5. On a refutation with assumptions, add these refuted assumptions to the SAT solver and recompute the model

AVATAR Architecture

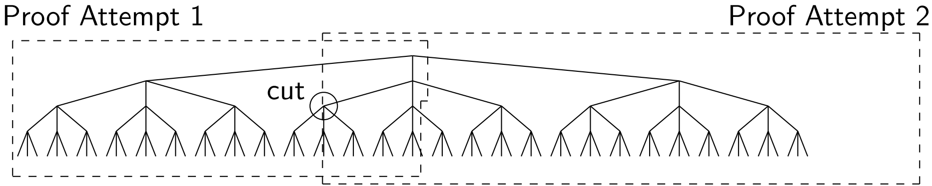


Communicating Splitting Decisions

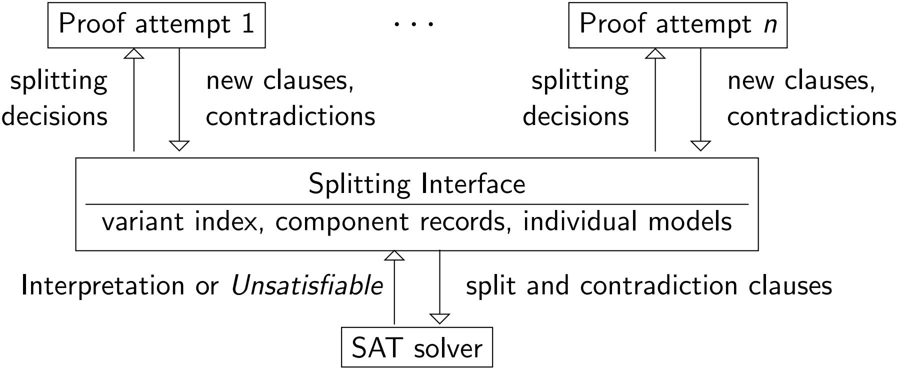
- Idea: if one proof attempt shows a part of the splitting space to be inconsistent then another proof attempt doesn't need to explore it
- Very easy to share such splitting decisions via AVATAR - just share the SAT solver
- Has the effect of allowing proof attempts to explore the search space much faster

Exploring the Search Space Together

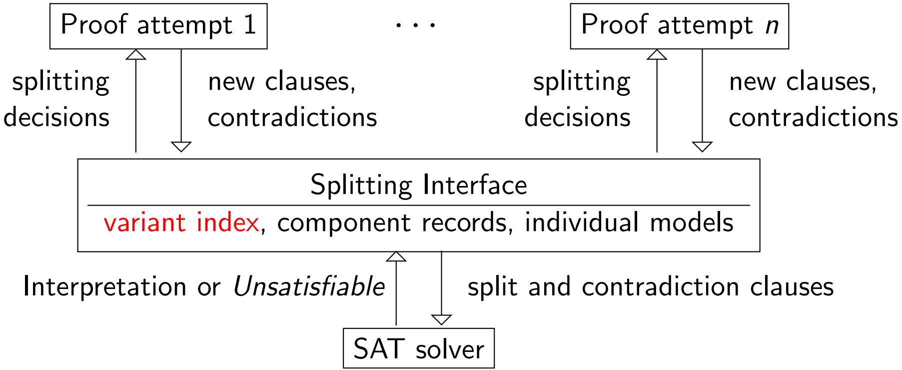
- Proof attempt 1 shows that assuming a component of a clause leads to contradiction
- Proof attempt 2 can ignore any splitting branch containing this component



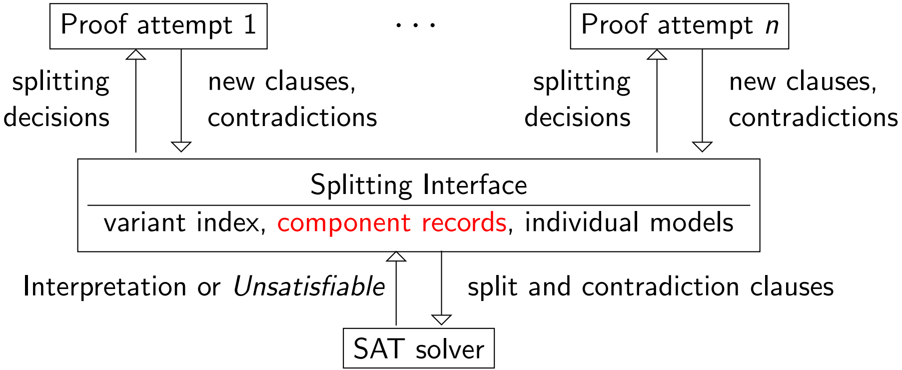
Shared AVATAR Architecture



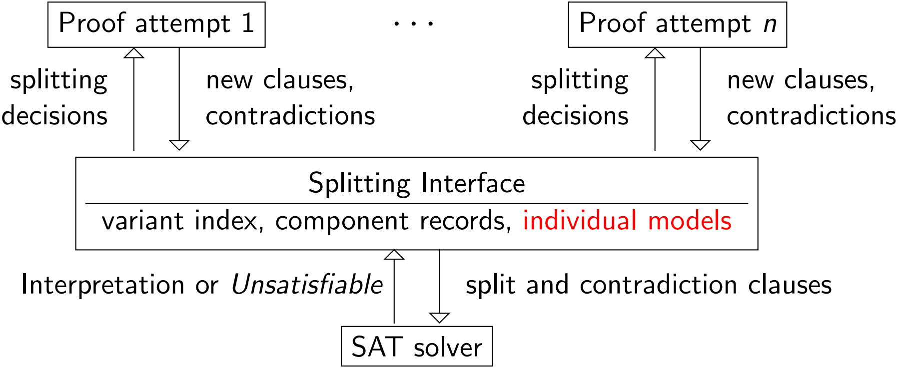
Shared AVATAR Architecture



Shared AVATAR Architecture



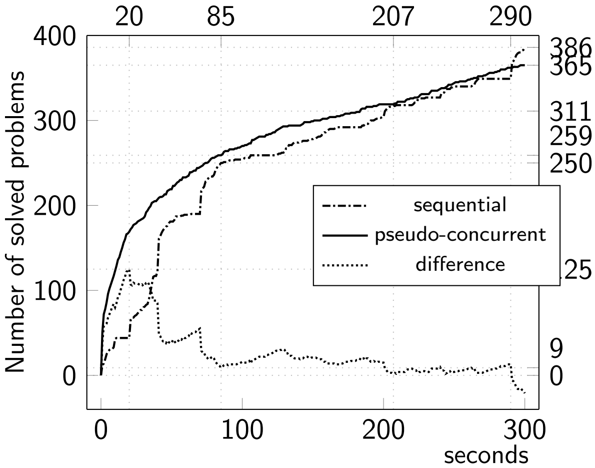
Shared AVATAR Architecture



Experiment

- We took
  - 1747 very hard first-order problems from TPTP
  - 30 random 'sensible' strategies
- And ran
  - Each strategy independently for 10 seconds
  - All 30 together with a per-strategy 10 second time limit
- We found
  - Problems were solved on average 1.53 times faster, in some cases it was much higher than this
  - Sharing splitting decisions led to 63 more problems being solved, often quickly. It also led to previously unsolved problems being solved - this is significant.
  - However some problems were lost. There are two explanations
    - SAT solver overhead goes up 20%
    - Loss of memory locality

Experiment


20
85
207
290
400
300
200
100
0
386
365
311
259
250
125
9
0
Number of solved problems
sequential
pseudo-concurrent
difference
0
100
200
300
seconds

Replacing the SAT solver with a SMT solver

- A big advantage of this architecture is that we can replace the SAT solver with a SMT solver and only search models that satisfy some set of theories
- This only requires ground components to be passed directly instead of being represented by a SAT variable
- We are currently experimenting with incorporating Z3 for this purpose and the results are ~~encouraging~~ good

Conclusions

- A very promising direction to prove more problems and prove them faster
- Plugging in a SMT solver will make this approach highly applicable to problems with quantifiers and theories
- Still lots of ways we can extend the architecture i.e. cooperating via other data structures
- Some engineering problems still to solve

Thank you for listening