Playing with Vampire: the dark art of theorem proving

Martin Suda

TU Wien, Vienna, Austria

SAT/SMT/AR Summer School in Manchester, 2018

Vampire is . . .

. . . an automated theorem prover for first-order logic

Vampire is . . .

. . . an automated theorem prover for first-order logic

What is special about Vampire?

- very fast (35 trophies from CASC over the last 16 years)

Vampire is . . .

. . . an automated theorem prover for first-order logic

What is special about Vampire?

- very fast (35 trophies from CASC over the last 16 years)
- ~~notoriously hard to obtain~~

Vampire is . . .

. . . an automated theorem prover for first-order logic

What is special about Vampire?

- very fast (35 trophies from CASC over the last 16 years)
- ~~notoriously hard to obtain~~
- simple to use, but also many ways to configure

Vampire is . . .

. . . an automated theorem prover for first-order logic

What is special about Vampire?

- very fast (35 trophies from CASC over the last 16 years)
- ~~notoriously hard to obtain~~
- simple to use, but also many ways to configure
- fully TPTP compliant; also understands SMTLIB2

Vampire is . . .

. . . an automated theorem prover for first-order logic

What is special about Vampire?

- very fast (35 trophies from CASC over the last 16 years)
- ~~notoriously hard to obtain~~
- simple to use, but also many ways to configure
- fully TPTP compliant; also understands SMTLIB2
- detailed proofs

Vampire is . . .

. . . an automated theorem prover for first-order logic

What is special about Vampire?

- very fast (35 trophies from CASC over the last 16 years)
- ~~notoriously hard to obtain~~
- simple to use, but also many ways to configure
- fully TPTP compliant; also understands SMTLIB2
- detailed proofs
- limited resource strategy (LRS)

Vampire is . . .

. . . an automated theorem prover for first-order logic

What is special about Vampire?

- very fast (35 trophies from CASC over the last 16 years)
- ~~notoriously hard to obtain~~
- simple to use, but also many ways to configure
- fully TPTP compliant; also understands SMTLIB2
- detailed proofs
- limited resource strategy (LRS)
- AVATAR architecture for clause splitting

Vampire is . . .

. . . an automated theorem prover for first-order logic

What is special about Vampire?

- very fast (35 trophies from CASC over the last 16 years)
- ~~notoriously hard to obtain~~
- simple to use, but also many ways to configure
- fully TPTP compliant; also understands SMTLIB2
- detailed proofs
- limited resource strategy (LRS)
- AVATAR architecture for clause splitting
- reasoning with theories (arithmetic, arrays, . . .) and quantifiers (!)

Vampire is . . .

. . . an automated theorem prover for first-order logic

What is special about Vampire?

- very fast (35 trophies from CASC over the last 16 years)
- ~~notoriously hard to obtain~~
- simple to use, but also many ways to configure
- fully TPTP compliant; also understands SMTLIB2
- detailed proofs
- limited resource strategy (LRS)
- AVATAR architecture for clause splitting
- reasoning with theories (arithmetic, arrays, . . .) and quantifiers (!)
- finite model building

Vampire is . . .

. . . an automated theorem prover for first-order logic

What is special about Vampire?

- very fast (35 trophies from CASC over the last 16 years)
- ~~notoriously hard to obtain~~
- simple to use, but also many ways to configure
- fully TPTP compliant; also understands SMTLIB2
- detailed proofs
- limited resource strategy (LRS)
- AVATAR architecture for clause splitting
- reasoning with theories (arithmetic, arrays, . . .) and quantifiers (!)
- finite model building
- powerful portfolio mode, a.k.a "the CASC mode"

Vampire is . . .

. . . an automated theorem prover for first-order logic

What is special about Vampire?

- very fast (35 trophies from CASC over the last 16 years)
- ~~notoriously hard to obtain~~
- simple to use, but also many ways to configure
- fully TPTP compliant; also understands SMTLIB2
- detailed proofs
- limited resource strategy (LRS)
- AVATAR architecture for clause splitting
- reasoning with theories (arithmetic, arrays, . . .) and quantifiers (!)
- finite model building
- powerful portfolio mode, a.k.a "the CASC mode"
- question answering, interpolants, sine axiom selection, . . .

CASC 2017 results[1]

Higher-order Theorems	Satallax 3.2	Leo-III 1.1	Satallax 3.0	LEO-II 1.7.0	Zipperpin 1.1	Isabelle 2016
Solved/500	430/500	382/500	382/500	305/500	179/500	387/500
Solutions	430/500	382/500	375/500	301/500	179/500	0/500

Typed First-order Theorems +*-/	Vampire 4.1	Vampire 4.2	CVC4 ARI-1.5.2	Princess 170717	Zipperpin 1.1
Solved/250	194/250	191/250	188/250	130/250	39/250
Solutions	194/250	191/250	188/250	115/250	39/250

First-order Theorems	Vampire 4.2	Vampire 4.0	E 2.1	CVC4 NAR-1.5.2	iProver 2.6	Leo-III 1.1	lean-nan 1.0	Zipperpin 1.1
Solved/500	452/500	444/500	381/500	327/500	283/500	211/500	186/500	154/500
Solutions	452/500	440/500	381/500	327/500	279/500	211/500	186/500	154/500

First-order Non-theorems	Vampire SAT-4.1	Vampire SAT-4.2	iProver SAT-2.6	CVC4 SNA-1.5.2	E FNT-2.1	Scavenger EP-0.2
Solved/250	219/250	217/250	175/250	136/250	85/250	12/250
Solutions	217/250	204/250	175/250	136/250	85/250	12/250

Effectively Propositional CNF	iProver 2.6	iProver 2.5	Vampire 4.2	E 2.1	Scavenger EP-0.1	Scavenger EP-0.2
Solved/200	174/200	171/200	168/200	53/200	5/200	4/200

SLedgeHammer Theorems	Vampire SLH-4.2	CVC4 SLH-1.5.2	ET 2.0	E SLH-2.1	Leo-III SLH-1.1	iProver SLH-2.6	Zipperpin SLH-1.1	iProverM 2.5-0.1
Solved/2000	1433/2000	1364/2000	1328/2000	1185/2000	652/2000	519/2000	472/2000	320/2000

Large Theory Batch Problems	Vampire LTB-4.0	Vampire LTB-4.2	MaLARe 0.6	iProver LTB-2.6	E LTB-2.1
Solved/1500	1156/1500	1144/1486	1131/1500	777/1499	683/1499
Solutions	1156/1500	1144/1486	1131/1500	777/1499	683/1499

[1] http://www.cs.miami.edu/~tptp/CASC/26/WWWFiles/DivisionSummary1.html

Maybe you have seen in Kilburn building . . .

Obtaining Vampire and these slides

The source code of Vampire is available on GitHub:
`https://github.com/vprover/vampire`

Vampire has been pre-installed on the summer school VM

Like us  on Facebook:
`https://www.facebook.com/vprover/`

To get these slides and exercises:
`https://github.com/quickbeam123/satsmtar2018`

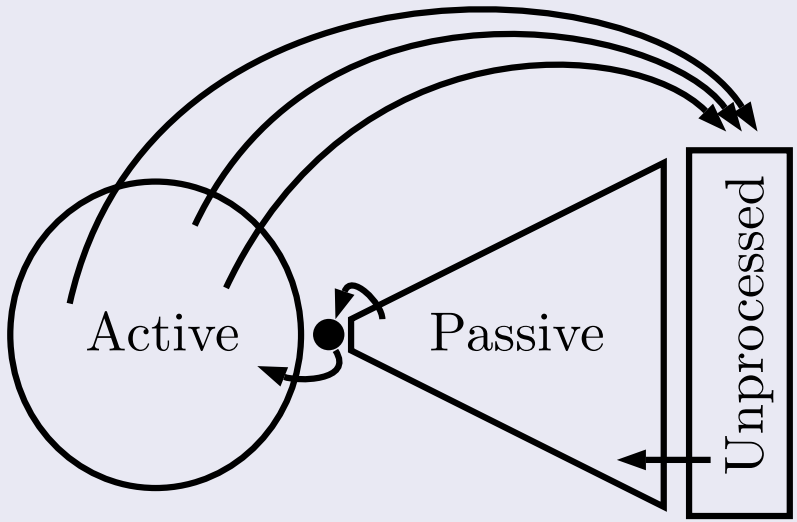
(`git clone https://github.com/quickbeam123/satsmtar2018.git`)

First-order logic and the TPTP langauge

Switching to the other slides for a bit . . .

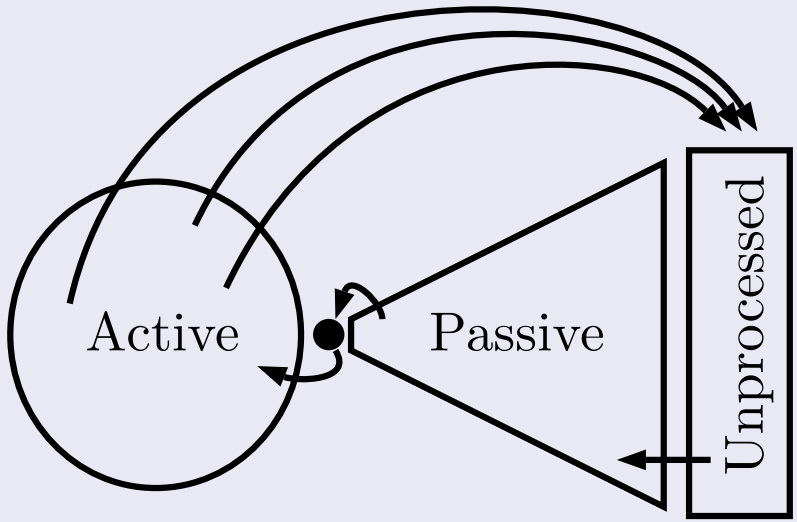
Saturation

Selecting the given clause



Saturation

Selecting the given clause

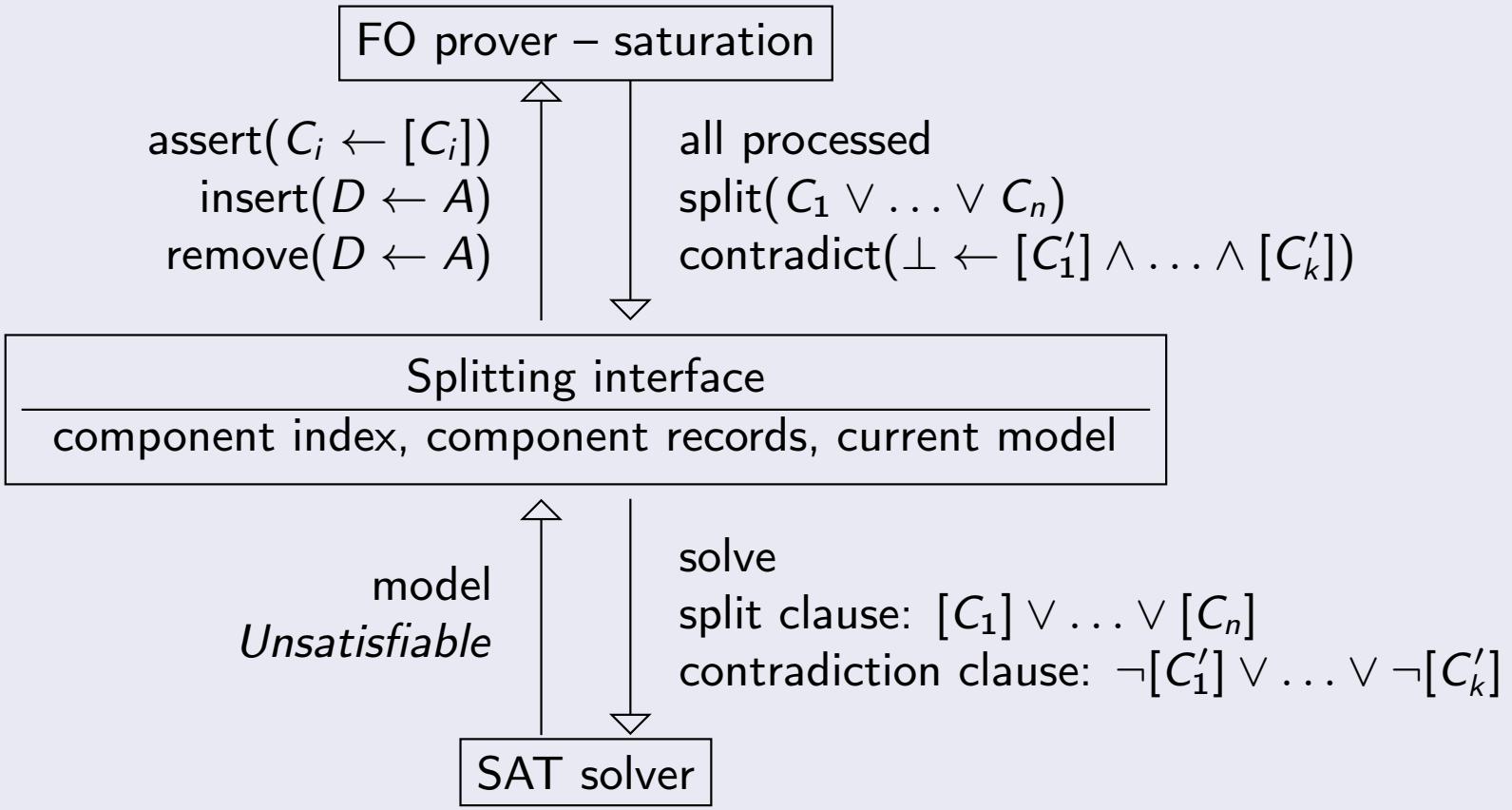


Saturation algorithms in Vampire

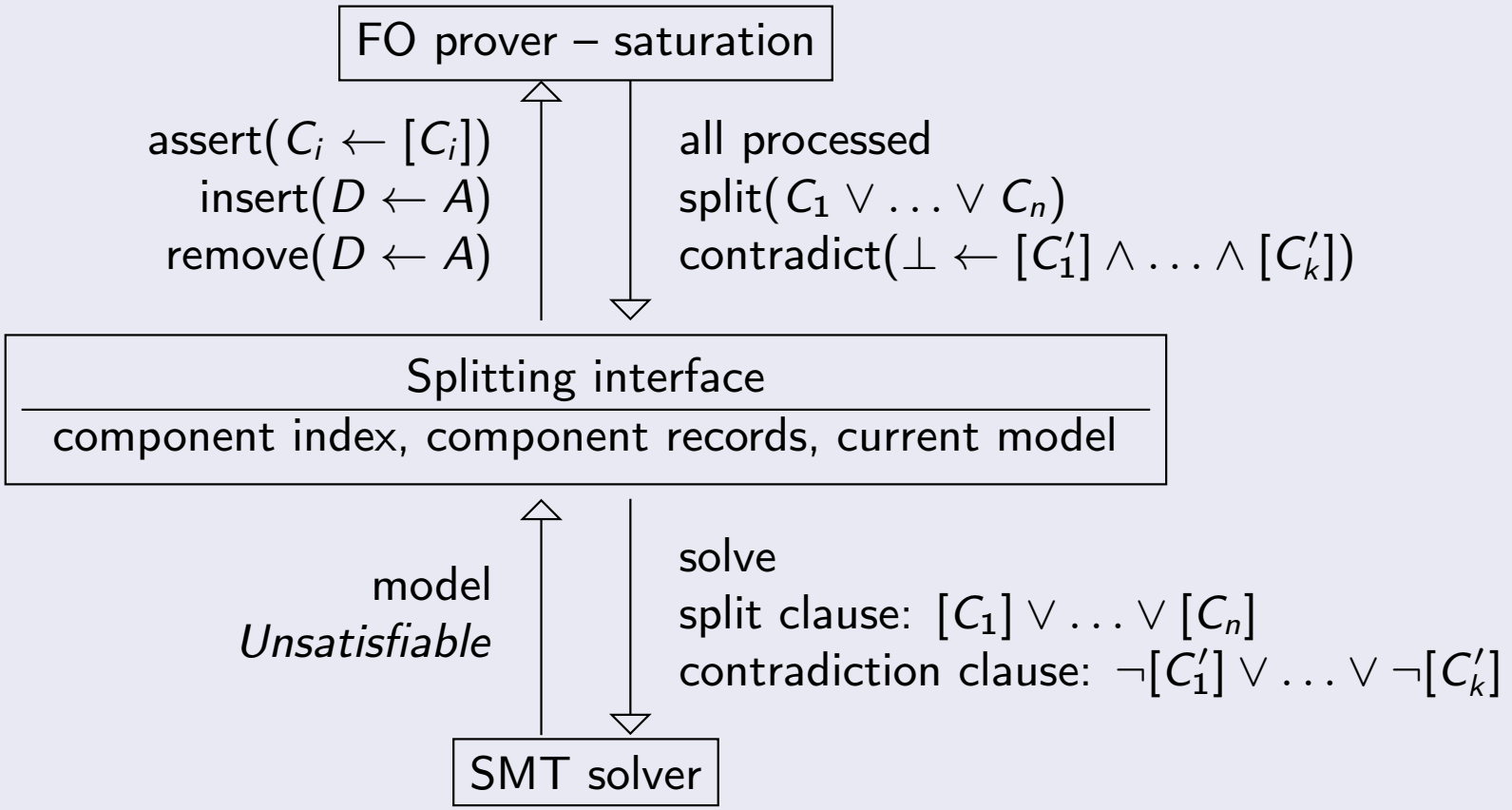
- the Discount loop
- the Otter loop
- Limited Resource Strategy [RV03]


V

AVATAR – architecture overview



AVATAR – architecture overview



Reasoning with theories

Arithmetic reasoning in Vampire

- Evaluation of ground interpreted terms ($1 + X \longrightarrow X$)
- Theory axioms
  - hand-crafted set
  - which axioms should be added for a given problem?
- VampireZ3 = AVATAR with an SMT solver
  - current implementation for Z3
  - Idea: Vampire only explores theory-consistent ground sub-problems
- theory instantiation and unification with abstraction [TACAS2018]
- ...

Reasoning with theories

Arithmetic reasoning in Vampire

- Evaluation of ground interpreted terms ($1 + X \longrightarrow X$)
- Theory axioms
  - hand-crafted set
  - which axioms should be added for a given problem?
- VampireZ3 = AVATAR with an SMT solver
  - current implementation for Z3
  - Idea: Vampire only explores theory-consistent ground sub-problems
- theory instantiation and unification with abstraction [TACAS2018]
- ...

```
tff(sum_something_0_samething,conjecture,(
    ! [X: $int] :
      ( ( $less(-1,X)
        & $less(X,1) )
     => $sum(21,X) = 21 ) )).

./vampire_z3_rel_master_3993 ARI163\=1.p
```

Finite model building

MACE/Paradox-style model finding

- try looking for a finite counterexample to an invalid conjecture
- iterate model sizes $n = 1, 2, \ldots$
- pose the question "$\exists M, |M| = n \;\&\; M \models Ax \wedge \neg C$" as a SAT problem

Finite model building

MACE/Paradox-style model finding

- try looking for a finite counterexample to an invalid conjecture
- iterate model sizes $n = 1, 2, \ldots$
- pose the question "$\exists M, |M| = n$ & $M \models Ax \wedge \neg C$" as a SAT problem

```
%---- 1 * x = x
fof(left_identity,axiom, ! [X] : mult(e,X) = X).
%---- i(x) * x = 1
fof(left_inverse,axiom, ! [X] : ?[Y] : mult(Y, X) = e).
%---- (x * y) * z = x * (y * z)
fof(associativity,axiom,
    ! [X,Y,Z] : mult(mult(X,Y),Z) = mult(X,mult(Y,Z))).

./vampire_z3_rel_master_3993 grp_ord2.p
```

Dark art mini-CHALLENGE

```
./vampire --show_options on
```

Dark art mini-CHALLENGE

`./vampire --show_options on`

Some options to play with

1. Set of support (`-sos on`)
   5536, but 507 new with respect to default
2. AVATAR turned off (`-av off`)
   default: 9465; avoff: 8587, also 221 new
3. Discount saturation loop and the age-weight ratio
   (`-sa discount -awr 10`)
   discount only: 9341; with awr10: 9495
4. Lookahead literal selection (`-s 1011`)
   default: 9465; lookahead: 8852 but 839 new
5. Backward subsumption (`-bs on`)

Ready made solution from the Vizzard

Portfolio mode (a.k.a. CASC mode)

- a conditional portfolio mode
- a cocktail of a strategies optimized for good general performance
- incomplete strategies in the mix; complementarity for coverage
- `--mode casc` (there is also `--mode casc_sat`)
- in fact: `--mode portfolio --schedule casc_2018 -t 5m`
- `--cores <number>` for executing in parallel

Ready made solution from the Vizzard

Portfolio mode (a.k.a. CASC mode)

- a conditional portfolio mode
- a cocktail of a strategies optimized for good general performance
- incomplete strategies in the mix; complementarity for coverage
- `--mode casc` (there is also `--mode casc_sat`)
- in fact: `--mode portfolio --schedule casc_2018 -t 5m`
- `--cores <number>` for executing in parallel

A small experiment (5 minutes time limit)

TPTP 7.0.0 total:	21851	
Discarded (hol + poly):	4323	
Eligible (cnf, fof, tff):	17528	
casc:	13362	76.2 %
casc_sat:	9473	54.0 %
union:	13495	76.9 %