Outline

Sorts and Theories
Sorts

Consider these statements:

1. Sort $b$ consists of two elements: $t$ and $f$;
2. Sort $s$ has three different elements.

\[
t! = f \land (\forall x : b)(x \cong t \lor x \cong f) \\
(\exists x : s)(\exists y : s)(\exists z : s)(x \not\cong y \land x \not\cong z \land y \not\cong z)
\]
Sorts

Consider these statements:

1. Sort $b$ consists of two elements: $t$ and $f$;
2. Sort $s$ has three different elements.

\[
t! = f \land (\forall x : b)(x \simeq t \lor x \simeq f)
(\exists x : s)(\exists y : s)(\exists z : s)(x \not\simeq y \land x \not\simeq z \land y \not\simeq z)
\]

The unsorted version of it:

\[
(\forall x)(x \simeq t \lor x \simeq f)
(\exists x)(\exists y)(\exists z)(x \not\simeq y \land x \not\simeq z \land y \not\simeq z)
\]

is unsatisfiable:

\[
\text{fof}(1, \text{axiom}, t \neq f \land \lnot [X] : X = t \lor X = f).
\text{fof}(1, \text{axiom}, ? [X,Y,Z] : (X \neq Y \land X \neq Z \land Y \neq Z)).
\]

\text{vampire} --splitting off
--saturation_algorithm inst_gen sort1.tptp
tff(boolean_type,type,b: $tType).  % b is a sort

\[ tff(s\_is\_a\_type,type,s: $tType).  \quad \% s is a sort \]

\[ tff(t\_has\_type\_b,type,t : b).  \quad \% t has sort b \]

\[ tff(f\_has\_type\_b,type,f : b).  \quad \% f has sort b \]

\[ tff(1,axiom,t \neq f \land \neg [X:b] : X = t \lor X = f). \]

\[ tff(1,axiom,\exists [X:s,Y:s,Z:s] : (X \neq Y \land X \neq Z \land Y \neq Z)). \]

vampire --splitting off

--saturation_algorithm inst_gen sort2.tptp
Pre-existing sorts

- $i$: sort of individuals. If is the default sort: if a symbol is not declared, it has this sort.
- $\text{int}$: sort of integers.
- $\text{rat}$: sort of rationals.
- $\text{real}$: sort of reals.
Integers

One can use concrete integers and some interpreted functions on them.

fof(1,conjecture,$\text{sum}(2,2)=4)$.

vampire --inequality_splitting 0 int1.tptp
Interpreted Functions and Predicates on Integers

Functions:

▶ $sum$: addition \((x + y)\)
▶ $product$: multiplication \((x \cdot y)\)
▶ $difference$: difference \((x - y)\)
▶ $uminus$: unary minus \((-x)\)
▶ $to\_rat$: conversion to rationals.
▶ $to\_real$: conversion to reals.

Predicates:

▶ $less$: less than \((x < y)\)
▶ $lesseq$: less than or equal to \((x \leq y)\)
▶ $greater$: greater than \((x > y)\)
▶ $greatereq$: greater than or equal to \((x \geq y)\)
How Vampire Proves Problems in Arithmetic

- adding theory axioms;
- evaluating expressions, when possible;
- (future) SMT solving.
How Vampire Proves Problems in Arithmetic

- adding theory axioms;
- evaluating expressions, when possible;
- (future) SMT solving.

Example:

\[(x + y) + z = x + (z + y)\].

```prolog
fof(1,conjecture, ![X:$int,Y:$int,Z:$int] : $sum($sum(X,Y),Z) =$sum(X,$sum(Z,Y))).
```

```bash
vampire --inequality_splitting 0 int2.tptp
```
How Vampire Proves Problems in Arithmetic

- adding theory axioms;
- evaluating expressions, when possible;
- (future) SMT solving.

Example:

\[(x + y) + z = x + (z + y)\].

```
fof(1,conjecture,
  ! [X:int,Y:int,Z:int] :
  \$sum($sum(X,Y),Z)=$sum(X,$sum(Z,Y))).
```

```
vampire --inequality_splitting 0 int2.tptp
```

- You can add your own axioms;
- you can replace Vampire axioms by your own: use
  ```
  --theory_axioms off
  ```