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.

$$t! = f \wedge (\forall x : b)(x \simeq t \vee x \simeq f)$$
$$(\exists x : s)(\exists y : s)(\exists z : s)(x \not\simeq y \wedge x \not\simeq z \wedge y \not\simeq z)$$

Sorts

Consider these statements:

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

$$t! = f \wedge (\forall x : b)(x \simeq t \vee x \simeq f)$$
$$(\exists x : s)(\exists y : s)(\exists z : s)(x \not\simeq y \wedge x \not\simeq z \wedge y \not\simeq z)$$

The unsorted version of it:

$$(\forall x)(x \simeq t \vee x \simeq f)$$
$$(\exists x)(\exists y)(\exists z)(x \not\simeq y \wedge x \not\simeq z \wedge y \not\simeq z)$$

is unsatisfiable:

```
fof(1,axiom,t != f & ! [X] : X = t | X = f).
fof(1,axiom,? [X,Y,Z] : (X != Y & X != Z & Y != Z)).
```

```
vampire --splitting off
  --saturation_algorithm inst_gen sort1.tptp
```

Sorts in TPTP

```
tff(boolean_type,type,b: $tType).    % b is a sort
tff(s_is_a_type,type,s: $tType).     % s is a sort

tff(t_has_type_b,type,t : b).    % t has sort b
tff(f_has_type_b,type,f : b).    % f has sort b

tff(1,axiom,t != f & ! [X:b] : X = t | X = f).
tff(1,axiom,? [X:s,Y:s,Z:s] : (X != Y & X != Z & Y != Z)).

vampire --splitting off
  --saturation_algorithm inst_gen sort2.tptp
```

Pre-existing sorts

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

Integers

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

```
fof(1,conjecture,$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:

$$(x + y) + z = x + (z + y).$$

```
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
```

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`