Outline

First-Order Logic and TPTP
First-Order Logic and TPTP

- **Language**: variables, function and predicate (relation) symbols. A constant symbol is a special case of a function symbol.
First-Order Logic and TPTP

- Language: variables, function and predicate (relation) symbols. A constant symbol is a special case of a function symbol. Variable names start with upper-case letters.
First-Order Logic and TPTP

- **Language**: variables, function and predicate (relation) symbols. A constant symbol is a special case of a function symbol. Variable names start with upper-case letters.

- **Terms**: variables, constants, and expressions $f(t_1, \ldots, t_n)$, where $f$ is a function symbol of arity $n$ and $t_1, \ldots, t_n$ are terms.
First-Order Logic and TPTP

- Language: variables, function and predicate (relation) symbols. A constant symbol is a special case of a function symbol. Variable names start with upper-case letters.

- Terms: variables, constants, and expressions $f(t_1, \ldots, t_n)$, where $f$ is a function symbol of arity $n$ and $t_1, \ldots, t_n$ are terms. Terms denote domain (universe) elements (objects).
First-Order Logic and TPTP

- Language: variables, function and predicate (relation) symbols. A constant symbol is a special case of a function symbol. Variable names start with upper-case letters.

- Terms: variables, constants, and expressions $f(t_1, \ldots, t_n)$, where $f$ is a function symbol of arity $n$ and $t_1, \ldots, t_n$ are terms. Terms denote domain (universe) elements (objects).

- Atomic formula: expression $p(t_1, \ldots, t_n)$, where $p$ is a predicate symbol of arity $n$ and $t_1, \ldots, t_n$ are terms.
First-Order Logic and TPTP

- **Language:** variables, function and predicate (relation) symbols. A constant symbol is a special case of a function symbol. Variable names start with upper-case letters.

- **Terms:** variables, constants, and expressions $f(t_1, \ldots, t_n)$, where $f$ is a function symbol of arity $n$ and $t_1, \ldots, t_n$ are terms. Terms denote domain (universe) elements (objects).

- **Atomic formula:** expression $p(t_1, \ldots, t_n)$, where $p$ is a predicate symbol of arity $n$ and $t_1, \ldots, t_n$ are terms. Formulas denote properties of domain elements.

- All symbols are uninterpreted, apart from equality $\equiv$. 
First-Order Logic and TPTP

- **Language**: variables, function and predicate (relation) symbols. A constant symbol is a special case of a function symbol. Variable names start with upper-case letters.

- **Terms**: variables, constants, and expressions $f(t_1, \ldots, t_n)$, where $f$ is a function symbol of arity $n$ and $t_1, \ldots, t_n$ are terms. Terms denote domain (universe) elements (objects).

- **Atomic formula**: expression $p(t_1, \ldots, t_n)$, where $p$ is a predicate symbol of arity $n$ and $t_1, \ldots, t_n$ are terms. Formulas denote properties of domain elements.

- All symbols are uninterpreted, apart from equality $\equiv$.

<table>
<thead>
<tr>
<th>FOL</th>
<th>TPTP</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\bot, T$</td>
<td>$$false,$$true $</td>
</tr>
<tr>
<td>$\neg F$</td>
<td>$\sim F$</td>
</tr>
<tr>
<td>$F_1 \wedge \ldots \wedge F_n$</td>
<td>$F_1 &amp; \ldots &amp; F_n$</td>
</tr>
<tr>
<td>$F_1 \lor \ldots \lor F_n$</td>
<td>$F_1</td>
</tr>
<tr>
<td>$F_1 \rightarrow F_n$</td>
<td>$F_1 \Rightarrow F_n$</td>
</tr>
<tr>
<td>$(\forall x_1) \ldots (\forall x_n)F$</td>
<td>![ [X1, ..., Xn] : F ]</td>
</tr>
<tr>
<td>$(\exists x_1) \ldots (\exists x_n)F$</td>
<td>![ [X1, ..., Xn] : F ]</td>
</tr>
</tbody>
</table>
More on the TPTP Syntax

%---- 1 * x = 1
fof(left_identity, axiom, (  
    ! [X] : mult(e, X) = X )).

%---- i(x) * x = 1
fof(left_inverse, axiom, (  
    ! [X] : mult(inverse(X), X) = e )).

%---- (x * y) * z = x * (y * z)
fof(associativity, axiom, (  
    ! [X,Y,Z] :  
        mult(mult(X,Y), Z) = mult(X, mult(Y, Z)) )).

%---- x * x = 1
fof(group_of_order_2, hypothesis,  
    ! [X] : mult(X, X) = e ).

%---- prove x * y = y * x
fof(commutativity, conjecture,  
    ! [X,Y] : mult(X, Y) = mult(Y, X) ).
More on the TPTP Syntax

- Comments;

```prolog
%---- 1 * x = 1
fof(left_identity, axiom, (
    ! [X] : mult(e, X) = X )).

%---- i(x) * x = 1
fof(left_inverse, axiom, (  
    ! [X] : mult(inverse(X), X) = e )).

%---- (x * y) * z = x * (y * z)
fof(associativity, axiom, (  
    ! [X,Y,Z] :  
        mult(mult(X,Y), Z) = mult(X, mult(Y, Z)) )).

%---- x * x = 1
fof(group_of_order_2, hypothesis, (  
    ! [X] : mult(X, X) = e )).

%---- prove x * y = y * x
fof(commutativity, conjecture, (  
    ! [X,Y] : mult(X, Y) = mult(Y, X) ).
```
More on the TPTP Syntax

- Comments;
- Input formula names;

%---- 1 * x = 1
fof(left_identity, axiom, (
    ! [X] : mult(e, X) = X ).

%---- i(x) * x = 1
fof(left_inverse, axiom, (
    ! [X] : mult(inverse(X), X) = e ).

%---- (x * y) * z = x * (y * z)
fof(associativity, axiom, (
    ! [X, Y, Z] : 
        mult(mult(X, Y), Z) = mult(X, mult(Y, Z) ).

%---- x * x = 1
fof(group_of_order_2, hypothesis, 
    ! [X] : mult(X, X) = e ).

%---- prove x * y = y * x
fof(commutativity, conjecture, 
    ! [X, Y] : mult(X, Y) = mult(Y, X) ).
More on the TPTP Syntax

- **Comments**;
- **Input formula names**;
- **Input formula roles** (very important);

%---- 1 * x = 1
fof(left_identity,axiom,(
    ! [X] : mult(e,X) = X )).

%---- i(x) * x = 1
fof(left_inverse,axiom,(
    ! [X] : mult(inverse(X),X) = e )).

%---- (x * y) * z = x * (y * z)
fof(associativity,axiom,(
    ! [X,Y,Z] : 
    mult(mult(X,Y),Z) = mult(X,mult(Y,Z)) )).

%---- x * x = 1
fof(group_of_order_2,hypothesis,(
    ! [X] : mult(X,X) = e )).

%---- prove x * y = y * x
fof(commutativity,conjecture,(
    ! [X,Y] : mult(X,Y) = mult(Y,X) ).
More on the TPTP Syntax

- Comments;
- Input formula names;
- Input formula roles (very important);
- Equality

%---- 1 * x = 1
fof(left_identity,axiom,( ! [X] : mult(e,X) = X ))).

%---- i(x) * x = 1
fof(left_inverse,axiom,( ! [X] : mult(inverse(X),X) = e ))).

%---- (x * y) * z = x * (y * z)
fof(associativity,axiom,( ! [X,Y,Z] :
    mult(mult(X,Y),Z) = mult(X,mult(Y,Z)) )).

%---- x * x = 1
fof(group_of_order_2,hypothesis,
    ! [X] : mult(X,X) = e ).

%---- prove x * y = y * x
fof(commutativity,conjecture,
    ! [X,Y] : mult(X,Y) = mult(Y,X) ).
Proof by Vampire (Slightly Modified)

Refutation found. Thanks to Tanya!
203. $false$ [subsumption resolution 202,14]
202. $sP1(mult(sK,sK0))$ [backward demodulation 188,15]
188. $mult(X8,X9) = mult(X9,X8)$ [superposition 22,87]
87. $mult(X2,mult(X1,X2)) = X1$ [forward demodulation 71,27]
71. $mult(inverse(X1),e) = mult(X2,mult(X1,X2))$ [superposition 23,20]
27. $mult(inverse(X2),e) = X2$ [superposition 22,10]
23. $mult(inverse(X4),mult(X4,X5)) = X5$ [forward demodulation 18,9]
22. $mult(X0,mult(X0,X1)) = X1$ [forward demodulation 16,9]
20. $e = mult(X0,mult(X1,mult(X0,X1)))$ [superposition 11,12]
18. $mult(e,X5) = mult(inverse(X4),mult(X4,X5))$ [superposition 11,10]
16. $mult(e,X1) = mult(X0,mult(X0,X1))$ [superposition 11,12]
15. $sP1(mult(sK0,sK))$ [inequality splitting 13,14]
14. $~sP1(mult(sK,sK0))$ [inequality splitting name introduction]
13. $mult(sK,sK0) != mult(sK0,sK)$ [cnf transformation 8]
12. $e = mult(X0,X0) (0:5)$ [cnf transformation 4]
11. $mult(mult(X0,X1),X2)=mult(X0,mult(X1,X2))$ [cnf transformation 3]
10. $e = mult(inverse(X0),X0)$ [cnf transformation 2]
9. $mult(e,X0) = X0$ [cnf transformation 1]
8. $mult(sK,sK0) != mult(sK0,sK)$ [skolemisation 7]
7. $? [X0,X1] : mult(X0,X1) != mult(X1,X0)$ [ennf transformation 6]
6. $~! [X0,X1] : mult(X0,X1) = mult(X1,X0)$ [negated conjecture 5]
5. $! [X0,X1] : mult(X0,X1) = mult(X1,X0)$ [input]
4. $! [X0] : e = mult(X0,X0)$ [input]
3. $! [X0,X1,X2] : mult(mult(X0,X1),X2) = mult(X0,mult(X1,X2))$ [input]
2. $! [X0] : e = mult(inverse(X0),X0)$ [input]
1. $! [X0] : mult(e,X0) = X0$ [input]
Proof by Vampire (Slightly Modified)

Refutation found. Thanks to Tanya!
203. $false$ [subsumption resolution 202,14]
202. $sP1(mult(sK,sK0))$ [backward demodulation 188,15]
188. $mult(X8,X9) = mult(X9,X8)$ [superposition 22,87]
87. $mult(X2,mult(X1,X2)) = X1$ [forward demodulation 71,27]
71. $mult(inverse(X1),e) = mult(X2,mult(X1,X2))$ [superposition 23,20]
27. $mult(inverse(X2),e) = X2$ [superposition 22,10]
23. $mult(inverse(X4),mult(X4,X5)) = X5$ [forward demodulation 18,9]
22. $mult(X0,mult(X0,X1)) = X1$ [forward demodulation 16,9]
20. $e = mult(X0,mult(X1,mult(X0,X1)))$ [superposition 11,12]
18. $mult(e,X5) = mult(inverse(X4),mult(X4,X5))$ [superposition 11,10]
16. $mult(e,X1) = mult(X0,mult(X0,X1))$ [superposition 11,12]
15. $sP1(mult(sK0,sK))$ [inequality splitting 13,14]
14. $\neg sP1(mult(sK,sK0))$ [inequality splitting name introduction]
13. $mult(sK,sK0) \neq mult(sK0,sK)$ [cnf transformation 8]
12. $e = mult(X0,X0)$ (0:5) [cnf transformation 4]
11. $mult(mult(X0,X1),X2)=mult(X0,mult(X1,X2))$ [cnf transformation 3]
10. $e = mult(inverse(X0),X0)$ [cnf transformation 2]
  9. $mult(e,X0) = X0$ [cnf transformation 1]
  8. $mult(sK,sK0) \neq mult(sK0,sK)$ [skolemisation 7]
  7. $\neg P1(X0,X1) : mult(X0,X1) \neq mult(X1,X0)$ [ennf transformation 6]
  6. $\neg P1(X0,X1) : mult(X0,X1) = mult(X1,X0)$ [negated conjecture 5]
  5. $P1(X0,X1) : mult(X0,X1) = mult(X1,X0)$ [input]
  4. $P1(X0) : e = mult(X0,X0)$ [input]
  3. $P1(X0,X1,X2) : mult(mult(X0,X1),X2) = mult(X0,mult(X1,X2))$ [input]
  2. $P1(X0) : e = mult(inverse(X0),X0)$ [input]
  1. $P1(X0) : mult(e,X0) = X0$ [input]

▶ Each inference derives a formula from zero or more other formulas;
Proof by Vampire (Slightly Modified)

Refutation found. Thanks to Tanya!

203. $false \ [\text{subsumption resolution 202,14}]
202. sP1(mult(sK,sK0)) \ [\text{backward demodulation 188,15}]
188. mult(X8,X9) = mult(X9,X8) \ [\text{superposition 22,87}]
87. mult(X2,mult(X1,X2)) = X1 \ [\text{forward demodulation 71,27}]
71. mult(inverse(X1),e) = mult(X2,mult(X1,X2)) \ [\text{superposition 23,20}]
27. mult(inverse(X2),e) = X2 \ [\text{superposition 22,10}]
23. mult(inverse(X4),mult(X4,X5)) = X5 \ [\text{forward demodulation 18,9}]
22. mult(X0,mult(X0,X1)) = X1 \ [\text{forward demodulation 16,9}]
20. e = mult(X0,mult(X1,mult(X0,X1))) \ [\text{superposition 11,12}]
18. mult(e,X5) = mult(inverse(X4),mult(X4,X5)) \ [\text{superposition 11,10}]
16. mult(e,X1) = mult(X0,mult(X0,X1)) \ [\text{superposition 11,12}]
15. sP1(mult(sK0,sK)) \ [\text{inequality splitting 13,14}]
14. ¬sP1(mult(sK,sK0)) \ [\text{inequality splitting name introduction}]
13. mult(sK,sK0) \neq mult(sK0,sK) \ [\text{cnf transformation 8}]
12. e = mult(X0,X0) \ (0:5) \ [\text{cnf transformation 4}]
11. mult(mult(X0,X1),X2) = mult(X0,mult(X1,X2)) \ [\text{cnf transformation 3}]
10. e = mult(inverse(X0),X0) \ [\text{cnf transformation 2}]
9. mult(e,X0) = X0 \ [\text{cnf transformation 1}]
8. mult(sK,sK0) \neq mult(sK0,sK) \ [\text{skolemisation 7}]
7. ? \ [X0,X1] : mult(X0,X1) \neq mult(X1,X0) \ [\text{ennf transformation 6}]
6. ¬! \ [X0,X1] : mult(X0,X1) = mult(X1,X0) \ [\text{negated conjecture 5}]
5. ! \ [X0,X1] : mult(X0,X1) = mult(X1,X0) \ [\text{input}]
4. ! \ [X0] : e = mult(X0,X0) \ [\text{input}]
3. ! \ [X0,X1,X2] : mult(mult(X0,X1),X2) = mult(X0,mult(X1,X2)) \ [\text{input}]
2. ! \ [X0] : e = mult(inverse(X0),X0) \ [\text{input}]
1. ! \ [X0] : mult(e,X0) = X0 \ [\text{input}]

► Each inference derives a formula from zero or more other formulas;
► Input, preprocessing, new symbols introduction, superposition calculus
Refutation found. Thanks to Tanya!

203. $false \ [\text{subsumption resolution } 202,14]$

202. sP1(mult(sK,sK0)) \ [\text{backward demodulation } 188,15]$

188. mult(X8,X9) = mult(X9,X8) \ [\text{superposition } 22,87]$

87. mult(X2,mult(X1,X2)) = X1 \ [\text{forward demodulation } 71,27]$

71. mult(inverse(X1),e) = mult(X2,mult(X1,X2)) \ [\text{superposition } 23,20]$

27. mult(inverse(X2),e) = X2 \ [\text{superposition } 22,10]$

23. mult(inverse(X4),mult(X4,X5)) = X5 \ [\text{forward demodulation } 18,9]$

22. mult(X0,mult(X0,X1)) = X1 \ [\text{forward demodulation } 16,9]$

20. e = mult(X0,mult(X1,mult(X0,X1))) \ [\text{superposition } 11,12]$

18. mult(e,X5) = mult(inverse(X4),mult(X4,X5)) \ [\text{superposition } 11,10]$

16. mult(e,X1) = mult(X0,mult(X0,X1)) \ [\text{superposition } 11,12]$

15. sP1(mult(sK0,sK)) \ [\text{inequality splitting } 13,14]$

14. ¬sP1(mult(sK0,sK)) \ [\text{inequality splitting name introduction}]$

13. mult(sK,sK0) != mult(sK0,sK) \ [\text{cnf transformation } 8]$

12. e = mult(X0,X0) \ [\text{cnf transformation } 4]$

11. mult(mult(X0,X1),X2)=mult(X0,mult(X1,X2)) \ [\text{cnf transformation } 3]$

10. e = mult(inverse(X0),X0) \ [\text{cnf transformation } 2]$

9. mult(e,X0) = X0 \ [\text{cnf transformation } 1]$

8. mult(sK,sK0) != mult(sK0,sK) \ [\text{skolemisation } 7]$

7. ? [X0,X1] : mult(X0,X1) != mult(X1,X0) \ [\text{ennf transformation } 6]$

6. ¬! [X0,X1] : mult(X0,X1) = mult(X1,X0) \ [\text{negated conjecture } 5]$

5. ! [X0,X1] : mult(X0,X1) = mult(X1,X0) \ [\text{input}]$

4. ! [X0] : e = mult(X0,X0) \ [\text{input}]$

3. ! [X0,X1,X2] : mult(mult(X0,X1),X2) = mult(X0,mult(X1,X2)) \ [\text{input}]$

2. ! [X0] : e = mult(inverse(X0),X0) \ [\text{input}]$

1. ! [X0] : mult(e,X0) = X0 \ [\text{input}]$

- Each inference derives a formula from zero or more other formulas;
- Input, preprocessing, new symbols introduction, superposition calculus
Refutation found. Thanks to Tanya!

203. \( \text{false} \) [subsumption resolution 202,14]
202. \( \text{sP1(mult(sK,sK0))} \) [backward demodulation 188,15]
188. \( \text{mult(X8,X9) = mult(X9,X8)} \) [superposition 22,87]
87. \( \text{mult(X2,mult(X1,X2)) = X1} \) [forward demodulation 71,27]
71. \( \text{mult(inverse(X1),e) = mult(X2,mult(X1,X2))} \) [superposition 23,20]
27. \( \text{mult(inverse(X2),e) = X2} \) [superposition 22,10]
23. \( \text{mult(inverse(X4),mult(X4,X5)) = X5} \) [forward demodulation 18,9]
22. \( \text{mult(X0,mult(X0,X1)) = X1} \) [forward demodulation 16,9]
20. \( \text{e = mult(X0,mult(X1,mult(X0,X1)))} \) [superposition 11,12]
18. \( \text{mult(e,X5) = mult(inverse(X4),mult(X4,X5))} \) [superposition 11,10]
16. \( \text{mult(e,X1) = mult(X0,mult(X0,X1))} \) [superposition 11,12]
15. \( \text{sP1(mult(sK0,sK))} \) [inequality splitting 13,14]
14. \( \neg \text{sP1(mult(sK,sK0))} \) [inequality splitting name introduction]
13. \( \text{mult(sK,sK0) != mult(sK0,sK)} \) [cnf transformation 8]
12. \( \text{e = mult(X0,X0) (0:5)} \) [cnf transformation 4]
11. \( \text{mult(mult(X0,X1),X2) = mult(X0,mult(X1,X2))} \) [cnf transformation 3]
10. \( \text{e = mult(inverse(X0),X0)} \) [cnf transformation 2]
9. \( \text{mult(e,X0) = X0} \) [cnf transformation 1]
8. \( \text{mult(sK,sK0) != mult(sK0,sK)} \) [skolemisation 7]
7. \( \neg \exists [X0,X1] : \text{mult(X0,X1) != mult(X1,X0)} \) [ennf transformation 6]
6. \( \neg \exists [X0,X1] : \text{mult(X0,X1) = mult(X1,X0)} \) [negated conjecture 5]
5. \( \exists [X0,X1] : \text{mult(X0,X1) = mult(X1,X0)} \) [input]
4. \( \exists [X0] : \text{e = mult(X0,X0)} \) [input]
3. \( \exists [X0,X1,X2] : \text{mult(mult(X0,X1),X2) = mult(X0,mult(X1,X2))} \) [input]
2. \( \exists [X0] : \text{e = mult(inverse(X0),X0)} \) [input]
1. \( \exists [X0] : \text{mult(e,X0) = X0} \) [input]

- Each inference derives a formula from zero or more other formulas;
- Input, preprocessing, new symbols introduction, superposition calculus
Proof by Vampire (Slightly Modified)

Refutation found. Thanks to Tanya!

203. $false$ [subsumption resolution 202,14]
202. $sP1(mult(sK,sK0))$ [backward demodulation 188,15]
188. mult($X8,X9) = mult(X9,X8)$ [superposition 22,87]
87. mult($X2,mult(X1,X2)) = X1$ [forward demodulation 71,27]
71. mult($inverse(X1),e) = mult(X2,mult(X1,X2))$ [superposition 23,20]
27. mult($inverse(X2),e) = X2$ [superposition 22,10]
23. mult($inverse(X4),mult(X4,X5)) = X5$ [forward demodulation 18,9]
22. mult($X0,mult(X0,X1)) = X1$ [forward demodulation 16,9]
20. $e = mult(X0,mult(X1,mult(X0,X1)))$ [superposition 11,12]
18. mult($e,X5) = mult(inverse(X4),mult(X4,X5))$ [superposition 11,10]
16. mult($e,X1) = mult(X0,mult(X0,X1))$ [superposition 11,12]
15. $sP1(mult(sK0,sK))$ [inequality splitting 13,14]
14. $\sim sP1(mult(sK,sK0))$ [inequality splitting name introduction]
13. mult($sK,sK0) != mult(sK0,sK)$ [cnf transformation 8]
12. $e = mult(X0,X0) (0:5)$ [cnf transformation 4]
11. mult($mult(X0,X1),X2) = mult(X0,mult(X1,X2))$ [cnf transformation 3]
10. $e = mult(inverse(X0),X0)$ [cnf transformation 2]
 9. mult($e,X0) = X0$ [cnf transformation 1]
 8. mult($sK,sK0) != mult(sK0,sK)$ [skolemisation 7]
 7. $\sim [X0,X1] : mult(X0,X1) != mult(X1,X0)$ [ennf transformation 6]
 6. $! [X0,X1] : mult(X0,X1) = mult(X1,X0)$ [negated conjecture 5]
 5. $! [X0,X1] : mult(X0,X1) = mult(X1,X0)$ [input]
 4. $! [X0] : e = mult(X0,X0)$ [input]
 3. $! [X0,X1,X2] : mult(mult(X0,X1),X2) = mult(X0,mult(X1,X2))$ [input]
 2. $! [X0] : e = mult(inverse(X0),X0)$ [input]
 1. $! [X0] : mult(e,X0) = X0$ [input]

- Each inference derives a formula from zero or more other formulas;
- Input, preprocessing, new symbols introduction, superposition calculus
Refutation found. Thanks to Tanya!

203. $false$ [subsumption resolution 202,14]
202. sP1(mult(sK,sK0)) [backward demodulation 188,15]
188. mult(X8,X9) = mult(X9,X8) [superposition 22,87]
 87. mult(X2,mult(X1,X2)) = X1 [forward demodulation 71,27]
 71. mult(inverse(X1),e) = mult(X2,mult(X1,X2)) [superposition 23,20]
27. mult(inverse(X2),e) = X2 [superposition 22,10]
23. mult(inverse(X4),mult(X4,X5)) = X5 [forward demodulation 18,9]
22. mult(X0,mult(X0,X1)) = X1 [forward demodulation 16,9]
20. e = mult(X0,mult(X1,mult(X0,X1))) [superposition 11,12]
18. mult(e,X5) = mult(inverse(X4),mult(X4,X5)) [superposition 11,10]
16. mult(e,X1) = mult(X0,mult(X0,X1)) [superposition 11,12]
15. sP1(mult(sK0,sK)) [inequality splitting 13,14]
14. ~sP1(mult(sK,sK0)) [inequality splitting name introduction]
13. mult(sK,sK0) != mult(sK0,sK) [cnf transformation 8]
12. e = mult(X0,X0) (0:5) [cnf transformation 4]
11. mult(mult(X0,X1),X2) = mult(X0,mult(X1,X2)) [cnf transformation 3]
10. e = mult(inverse(X0),X0) [cnf transformation 2]
 9. mult(e,X0) = X0 [cnf transformation 1]
 8. mult(sK,sK0) != mult(sK0,sK) [skolemisation 7]
 7. ? [X0,X1] : mult(X0,X1) != mult(X1,X0) [ennf transformation 6]
 6. ~! [X0,X1] : mult(X0,X1) = mult(X1,X0) [negated conjecture 5]
 5. ! [X0,X1] : mult(X0,X1) = mult(X1,X0) [input]
 4. ! [X0] : e = mult(X0,X0) [input]
 3. ! [X0,X1,X2] : mult(mult(X0,X1),X2) = mult(X0,mult(X1,X2)) [input]
 2. ! [X0] : e = mult(inverse(X0),X0) [input]
 1. ! [X0] : mult(e,X0) = X0 [input]

▶ Each inference derives a formula from zero or more other formulas;
▶ Input, preprocessing, new symbols introduction, superposition calculus
▶ Proof by refutation, generating and simplifying inferences, unused formulas . . .
Proof by Vampire (Slightly Modified)

Refutation found. Thanks to Tanya!
203. $false$ [subsumption resolution 202,14]
202. sP1(mult(sK,sK0)) [backward demodulation 188,15]
188. mult(X8,X9) = mult(X9,X8) [superposition 22,87]
87. mult(X2,mult(X1,X2)) = X1 [forward demodulation 71,27]
71. mult(inverse(X1),e) = mult(X2,mult(X1,X2)) [superposition 23,20]
27. mult(inverse(X2),e) = X2 [superposition 22,10]
23. mult(inverse(X4),mult(X4,X5)) = X5 [forward demodulation 18,9]
22. mult(X0,mult(X0,X1)) = X1 [forward demodulation 16,9]
20. e = mult(X0,mult(X1,mult(X0,X1))) [superposition 11,12]
18. mult(e,X5) = mult(inverse(X4),mult(X4,X5)) [superposition 11,10]
16. mult(e,X1) = mult(X0,mult(X0,X1)) [superposition 11,12]
15. sP1(mult(sK0,sK)) [inequality splitting 13,14]
14. ¬sP1(mult(sK,sK0)) [inequality splitting name introduction]
13. mult(sK,sK0) != mult(sK0,sK) [cnf transformation 8]
12. e = mult(X0,X0) (0:5) [cnf transformation 4]
11. mult(mult(X0,X1),X2)=mult(X0,mult(X1,X2)) [cnf transformation 3]
10. e = mult(inverse(X0),X0) [cnf transformation 2]
9. mult(e,X0) = X0 [cnf transformation 1]
8. mult(sK,sK0) != mult(sK0,sK) [skolemisation 7]
7. ? [X0,X1] : mult(X0,X1) != mult(X1,X0) [ennf transformation 6]
6. ¬! [X0,X1] : mult(X0,X1) = mult(X1,X0) [negated conjecture 5]
5. ! [X0,X1] : mult(X0,X1) = mult(X1,X0) [input]
4. ! [X0] : e = mult(X0,X0) [input]
3. ! [X0,X1,X2] : mult(mult(X0,X1),X2) = mult(X0,mult(X1,X2)) [input]
2. ! [X0] : e = mult(inverse(X0),X0) [input]
1. ! [X0] : mult(e,X0) = X0 [input]

- Each inference derives a formula from zero or more other formulas;
- Input, preprocessing, new symbols introduction, superposition calculus
- Proof by refutation, generating and simplifying inferences, unused formulas . . .
Refutation found. Thanks to Tanya!

203. \$false \text{ [subsumption resolution 202,14]}

202. sP1(mult(sK,sK0)) \text{ [backward demodulation 188,15]}

188. mult(X8,X9) = mult(X9,X8) \text{ [superposition 22,87]}

87. mult(X2,mult(X1,X2)) = X1 \text{ [forward demodulation 71,27]}

71. mult(inverse(X1),e) = mult(X2,mult(X1,X2)) \text{ [superposition 23,20]}

27. mult(inverse(X2),e) = X2 \text{ [superposition 22,10]}

23. mult(inverse(X4),mult(X4,X5)) = X5 \text{ [forward demodulation 18,9]}

22. mult(X0,mult(X0,X1)) = X1 \text{ [forward demodulation 16,9]}

20. e = mult(X0,mult(X1,mult(X0,X1))) \text{ [superposition 11,12]}

18. mult(e,X5) = mult(inverse(X4),mult(X4,X5)) \text{ [superposition 11,10]}

16. mult(e,X1) = mult(X0,mult(X0,X1)) \text{ [superposition 11,12]}

15. sP1(mult(sK0,sK)) \text{ [inequality splitting 13,14]}

14. \neg sP1(mult(sK,sK0)) \text{ [inequality splitting name introduction]}

13. mult(sK,sK0) \neq mult(sK0,sK) \text{ [cnf transformation 8]}

12. e = mult(X0,X0) \text{ (0:5) [cnf transformation 4]}

11. mult(mult(X0,X1),X2) = mult(X0,mult(X1,X2)) \text{ [cnf transformation 3]}

10. e = mult(inverse(X0),X0) \text{ [cnf transformation 2]}

9. mult(e,X0) = X0 \text{ [cnf transformation 1]}

8. mult(sK,sK0) \neq mult(sK0,sK) \text{ [skolemisation 7]}

7. \neg [X0,X1] : mult(X0,X1) \neq mult(X1,X0) \text{ [ennf transformation 6]}

6. \neg [X0,X1] : mult(X0,X1) = mult(X1,X0) \text{ [negated conjecture 5]}

5. [X0,X1] : mult(X0,X1) = mult(X1,X0) \text{ [input]}

4. [X0] : e = mult(X0,X0) \text{ [input]}

3. [X0,X1,X2] : mult(mult(X0,X1),X2) = mult(X0,mult(X1,X2)) \text{ [input]}

2. [X0] : e = mult(inverse(X0),X0) \text{ [input]}

1. [X0] : mult(e,X0) = X0 \text{ [input]}

- Each inference derives a formula from zero or more other formulas;
- Input, preprocessing, new symbols introduction, superposition calculus
- Proof by refutation, generating and simplifying inferences, unused formulas ...
Statistics

Version: Vampire 1.8 (revision 1362)
Termination reason: Refutation

Active clauses: 14
Passive clauses: 35
Generated clauses: 194
Final active clauses: 8
Final passive clauses: 11
Input formulas: 5
Initial clauses: 6

Splitted inequalities: 1

Fw subsumption resolutions: 1
Fw demodulations: 68
Bw demodulations: 14

Forward subsumptions: 65
Backward subsumptions: 1
Fw demodulations to eq. taut.: 20
Bw demodulations to eq. taut.: 1

Forward superposition: 60
Backward superposition: 39
Self superposition: 6

Unique components: 6

Memory used [KB]: 255
Time elapsed: 0.007 s
Vampire

- **Completely automatic:** once you started a proof attempt, it can only be interrupted by terminating the process.
Vampire

- **Completely automatic:** once you started a proof attempt, it can only be interrupted by terminating the process.
- Chris Weidenbach: *A dark side of theorem proving.*
Vampire

- Completely automatic: once you started a proof attempt, it can only be interrupted by terminating the process.
- Chris Weidenbach: A dark side of theorem proving.
- Anonymous referee: VAMPIRE is not a glass of Tuborg.
Main applications

- Software and hardware verification;
- Static analysis of programs;
- Query answering in first-order knowledge bases (ontologies);
- Theorem proving in mathematics, especially in algebra;
Main applications

- Software and hardware verification;
- Static analysis of programs;
- Query answering in first-order knowledge bases (ontologies);
- Theorem proving in mathematics, especially in algebra;
- Verification of cryptographic protocols;
- Retrieval of software components;
- Reasoning in non-classical logics;
- Program synthesis;
Main applications

- Software and hardware verification;
- Static analysis of programs;
- Query answering in first-order knowledge bases (ontologies);
- Theorem proving in mathematics, especially in algebra;
- Verification of cryptographic protocols;
- Retrieval of software components;
- Reasoning in non-classical logics;
- Program synthesis;
- Writing papers and giving talks at various conferences and schools . . .
What an Automatic Theorem Prover is Expected to Do

Input:
- a set of axioms (first order formulas) or clauses;
- a conjecture (first-order formula or set of clauses).

Output:
- proof (hopefully).
Proof by Refutation

Given a problem with axioms and assumptions $F_1, \ldots, F_n$ and conjecture $G$,

1. negate the conjecture;
2. establish unsatisfiability of the set of formulas $F_1, \ldots, F_n, \neg G$. 
Proof by Refutation

Given a problem with axioms and assumptions $F_1, \ldots, F_n$ and conjecture $G$,

1. negate the conjecture;
2. establish unsatisfiability of the set of formulas $F_1, \ldots, F_n, \neg G$.

Thus, we reduce the theorem proving problem to the problem of checking unsatisfiability.
Proof by Refutation

Given a problem with axioms and assumptions $F_1, \ldots, F_n$ and conjecture $G$,

1. negate the conjecture;
2. establish unsatisfiability of the set of formulas $F_1, \ldots, F_n, \neg G$.

Thus, we reduce the theorem proving problem to the problem of checking unsatisfiability.

In this formulation the negation of the conjecture $\neg G$ is treated like any other formula. In fact, Vampire (and other provers) internally treat conjectures differently, to make proof search more goal-oriented.
General Scheme (simplified)

- Read a problem;
- Determine proof-search options to be used for this problem;
- Preprocess the problem;
- Convert it into CNF;
- Run a saturation algorithm on it, try to derive $\bot$.
- If $\bot$ is derived, report the result, maybe including a refutation.
General Scheme (simplified)

- Read a problem;
- Determine proof-search options to be used for this problem;
- Preprocess the problem;
- Convert it into CNF;
- Run a saturation algorithm on it, try to derive ⊥.
- If ⊥ is derived, report the result, maybe including a refutation.

Trying to derive ⊥ using a saturation algorithm is the hardest part, which in practice may not terminate or run out of memory.