Question Answering Revis(it)ed Again

Martin Suda*

Czech Technical University in Prague, Czech Republic

The 8th Vampire Workshop, Nancy, July 2024

*Supported by project CORESENSE no. 101070254 under the Horizon Europe programme.

Question Answering in the TPTP World and in Vampire

Existential conjectures beg a question: What is it we proved exists?

```
fof(a,axiom,p(a)).
fof(b,axiom,p(b)).
fof(c,conjecture,?[X]:p(X)).
```

Question Answering in the TPTP World and in Vampire

Existential conjectures beg a question: What is it we proved exists?

```
fof(a,axiom,p(a)).
fof(b,axiom,p(b)).
fof(c,conjecture,?[X]:p(X)).
```

Answer Extraction for TPTP (by Sutcliffe, Stickel, Schulz, Urban)

`https://tptp.org/Proposals/AnswerExtraction.html`

Question Answering in the TPTP World and in Vampire

Existential conjectures beg a question: What is it we proved exists?

```
fof(a,axiom,p(a)).
fof(b,axiom,p(b)).
fof(c,conjecture,?[X]:p(X)).
```

Answer Extraction for TPTP (by Sutcliffe, Stickel, Schulz, Urban)

`https://tptp.org/Proposals/AnswerExtraction.html`

```
./vampire -qa plain q0_basic.p
% SZS answers Tuple [[b]|_] for q0_basic
```

Question Answering in the TPTP World and in Vampire

Existential conjectures beg a question: What is it we proved exists?

```
fof(a,axiom,p(a)).
fof(b,axiom,p(b)).
fof(c,question,?[X]:p(X)).
```

Answer Extraction for TPTP (by Sutcliffe, Stickel, Schulz, Urban)

`https://tptp.org/Proposals/AnswerExtraction.html`

```
./vampire -qa plain q0_basic.p
% SZS answers Tuple [[X->b]|_] for q0_basic
```

Question Answering in the TPTP World and in Vampire

Existential conjectures beg a question: What is it we proved exists?

```
fof(a,axiom,p(a)).
fof(b,axiom,p(b)).
fof(c,question,?[X]:p(X)).
```

Answer Extraction for TPTP (by Sutcliffe, Stickel, Schulz, Urban)

`https://tptp.org/Proposals/AnswerExtraction.html`

```
./vampire -qa plain q0_basic.p
% SZS answers Tuple [[X->b]|_] for q0_basic
```

Related to, but not the same as, Vampire's recent synthesis support

There is also `-qa synthesis`. Talk to Petra about that!

Answer Literals: Technology Behind QA

```
./vampire -qa plain questions/q0_basic.p
% Refutation found. Thanks to Tanya!
% SZS status Theorem for q0_basic
% SZS answers Tuple [[X->b]|_] for q0_basic
% SZS output start Proof for q0_basic
2.  p(b) [input(axiom)]
3.  ?[X0] : p(X0) [input(conjecture)]
4. ~?[X0] : p(X0) [negated conjecture 3]
5. ~?[X0] : (p(X0) & ans0(X0)) [answer literal injection 4]
6.  ![X0] : (~p(X0) | ~ans0(X0)) [ennf transformation 5]
8.  p(b) [cnf transformation 2]
9.  ~p(X0) | ~ans0(X0) [cnf transformation 6]
11. ~ans0(b) [resolution 9,8]
12. ans0(X0) [answer literal resolver]
13. $false [unit resulting resolution 11,12]
% SZS output end Proof for q0_basic
```

Answer Literals: Technology Behind QA

```
./vampire -qa plain questions/q0_basic.p
% Refutation found. Thanks to Tanya!
% SZS status Theorem for q0_basic
% SZS answers Tuple [[X->b]|_] for q0_basic
% SZS output start Proof for q0_basic
2.  p(b) [input(axiom)]
3.  ?[X0] : p(X0) [input(conjecture)]
4. ~?[X0] : p(X0) [negated conjecture 3]
5. ~?[X0] : (p(X0) & ans0(X0)) [answer literal injection 4]
6.  ![X0] : (~p(X0) | ~ans0(X0)) [ennf transformation 5]
8.  p(b) [cnf transformation 2]
9.  ~p(X0) | ~ans0(X0) [cnf transformation 6]
11. ~ans0(b) [resolution 9,8]
12. ans0(X0) [answer literal resolver]
13. $false [unit resulting resolution 11,12]
% SZS output end Proof for q0_basic
```

NB: Answer literals could potentially affect proof search.

Answers are actually Tuples

```
tff(a,question,?[X:$int,Y:$int]:$less(X,Y)).
```

Answers are actually Tuples

```
tff(a,question,?[X:$int,Y:$int]:$less(X,Y)).
```

➥ Theory reasoning can be part of the game!

Answers are actually Tuples

```
tff(a,question,?[X:$int,Y:$int]:$less(X,Y)).
```

➥ Theory reasoning can be part of the game!

```
% SZS answers Tuple [∀X0.[X->X0,Y->$sum(X0,1)]|_] for ...
```

Answers are actually Tuples

```
tff(a,question,?[X:$int,Y:$int]:$less(X,Y)).
```

➥ Theory reasoning can be part of the game!

```
% SZS answers Tuple [∀X0.[X->X0,Y->$sum(X0,1)]|_] for ...
```

➥ Answers may contain variables (any instance works).

Answers are actually Tuples

```
tff(a,question,?[X:$int,Y:$int]:$less(X,Y)).
```

➥ Theory reasoning can be part of the game!

```
% SZS answers Tuple [∀X0.[X->X0,Y->$sum(X0,1)]|_] for ...
```

➥ Answers may contain variables (any instance works).

```
./vampire -qa plain -qago on ...  # "ground only"
...
% SZS answers Tuple [[X->-1,Y->0]|_] for ...
...
```

Disjunctive Answers – Real World Example!

```
tff(q,question,?[X:$real,Y:$real]:
  (~$is_rat(X) & ~$is_rat(Y) & $is_rat(exp(X,Y)))).
```

Disjunctive Answers – Real World Example!

```
tff(q,question,?[X:$real,Y:$real]:
  (~$is_rat(X) & ~$is_rat(Y) & $is_rat(exp(X,Y)))).
```

```
~$is_rat(sqrt(2.0))

exp(exp(X,Y),Z)=exp(X,times(Y,Z))
times(sqrt(X),sqrt(X))=X
exp(sqrt(X),2.0)=X
```

Disjunctive Answers – Real World Example!

```
tff(q,question,?[X:$real,Y:$real]:
  (~$is_rat(X) & ~$is_rat(Y) & $is_rat(exp(X,Y)))).
```

```
~$is_rat(sqrt(2.0))

exp(exp(X,Y),Z)=exp(X,times(Y,Z))
times(sqrt(X),sqrt(X))=X
exp(sqrt(X),2.0)=X
```

```
% SZS answers Tuple [
   ([X->sqrt(2.0),Y->sqrt(2.0)]|
    [X->exp(sqrt(2.0),sqrt(2.0)),Y->sqrt(2.0)])|_] for ...
```

What Does a Disjunctive Answer Really Mean?

Model theoretically:

In every model M_i of the axioms there is an answer σ_i which makes the "naked conjecture" true (i.e., $M_i \models C\sigma_i$, for a conjecture $\exists x.C$).

What Does a Disjunctive Answer Really Mean?

Model theoretically:

In every model M_i of the axioms there is an answer σ_i which makes the "naked conjecture" true (i.e., $M_i \models C\sigma_i$, for a conjecture $\exists x.C$).

Proof theoretically (recall Herbrand's theorem):

"I only needed the reported instances of the conjecture to derive $\bot$" (and the rest followed by essentially propositional reasoning.)

What Does a Disjunctive Answer Really Mean?

Model theoretically:

In every model M_i of the axioms there is an answer σ_i which makes the "naked conjecture" true (i.e., $M_i \models C\sigma_i$, for a conjecture $\exists x.C$).

Proof theoretically (recall Herbrand's theorem):

"I only needed the reported instances of the conjecture to derive $\bot$" (and the rest followed by essentially propositional reasoning.)

➥ a different proof might requires a smaller disjunction

What Does a Disjunctive Answer Really Mean?

Model theoretically:

In every model M_i of the axioms there is an answer σ_i which makes the "naked conjecture" true (i.e., $M_i \models C\sigma_i$, for a conjecture $\exists x.C$).

Proof theoretically (recall Herbrand's theorem):

"I only needed the reported instances of the conjecture to derive $\bot$" (and the rest followed by essentially propositional reasoning.)

➥ a different proof might requires a smaller disjunction

So what about the extreme case?

```
fof(a,axiom,p). fof(b,axiom,~p). fof(q,question,?[X]:q(X)).
```

What Does a Disjunctive Answer Really Mean?

Model theoretically:

In every model M_i of the axioms there is an answer σ_i which makes the "naked conjecture" true (i.e., $M_i \models C\sigma_i$, for a conjecture $\exists x.C$).

Proof theoretically (recall Herbrand's theorem):

"I only needed the reported instances of the conjecture to derive $\bot$" (and the rest followed by essentially propositional reasoning.)

➥ a different proof might requires a smaller disjunction

So what about the extreme case?

```
fof(a,axiom,p). fof(b,axiom,~p). fof(q,question,?[X]:q(X)).
```

1. `% SZS answers Tuple [()|_] for ...`

What Does a Disjunctive Answer Really Mean?

Model theoretically:

In every model M_i of the axioms there is an answer σ_i which makes the "naked conjecture" true (i.e., $M_i \models C\sigma_i$, for a conjecture $\exists x.C$).

Proof theoretically (recall Herbrand's theorem):

"I only needed the reported instances of the conjecture to derive $\bot$" (and the rest followed by essentially propositional reasoning.)

➥ a different proof might requires a smaller disjunction

So what about the extreme case?

```
fof(a,axiom,p). fof(b,axiom,~p). fof(q,question,?[X]:q(X)).
```

1. `% SZS answers Tuple [()|_] for ...`
2. `% SZS answers Tuple [∀X0.[X->X0]|_] for ...`

What Does a Disjunctive Answer Really Mean?

Model theoretically:

In every model M_i of the axioms there is an answer σ_i which makes the "naked conjecture" true (i.e., $M_i \models C\sigma_i$, for a conjecture $\exists x.C$).

Proof theoretically (recall Herbrand's theorem):

"I only needed the reported instances of the conjecture to derive $\bot$" (and the rest followed by essentially propositional reasoning.)

➥ a different proof might requires a smaller disjunction

So what about the extreme case?

```
fof(a,axiom,p). fof(b,axiom,~p). fof(q,question,?[X]:q(X)).
```

1. `% SZS answers Tuple [()|_] for ...`
2. `% SZS answers Tuple [∀X0.[X->X0]|_] for ...`
3. `% SZS status ContradictoryAxioms for ...`

Where Does This Skolem Come From?

```
fof(a,axiom,![P]: ?[M]: motherOf(M,P)).
fof(b,axiom,![P]: ?[F]: fatherOf(F,P)).

fof(c,axiom,![X,Y]: (parentOf(X,Y) <=>
        motherOf(X,Y) | fatherOf(X,Y))).

fof(q,question,?[X]: parentOf(X,bob)).
```

Where Does This Skolem Come From?

```
fof(a,axiom,![P]: ?[M]: motherOf(M,P)).
fof(b,axiom,![P]: ?[F]: fatherOf(F,P)).

fof(c,axiom,![X,Y]: (parentOf(X,Y) <=>
        motherOf(X,Y) | fatherOf(X,Y))).

fof(q,question,?[X]: parentOf(X,bob)).


% SZS answers Tuple [[X->sK1(bob)]|_] for ...
%    sK1 introduced for X1 in
       1. ! [X0] : ? [X1] : motherOf(X1,X0) [input(axiom)]
```

Where Does This Skolem Come From?

```
fof(a,axiom,![P]: ?[M]: motherOf(M,P)).
fof(b,axiom,![P]: ?[F]: fatherOf(F,P)).

fof(c,axiom,![X,Y]: (parentOf(X,Y) <=>
        motherOf(X,Y) | fatherOf(X,Y))).

fof(q,question,?[X]: parentOf(X,bob)).
```

```
% SZS answers Tuple [[X->sK1(bob)]|_] for ...
%    sK1 introduced for X1 in
      1. ! [X0] : ? [X1] : motherOf(X1,X0) [input(axiom)]
```

Could support filtering out answers with Skolems . . .

. . . but note that sometimes this is all there is!

Why Stop With Existential Conjectures?

```
tff(q,question,![A:$int,B:$int]:?[R:$int]:
        $difference(A,R)=B).
```

Why Stop With Existential Conjectures?

```
tff(q,question,![A:$int,B:$int]:?[R:$int]:
        $difference(A,R)=B).

% SZS status Theorem for ...
% SZS answers Tuple
        [λA,B.[R->$uminus($sum($uminus(A),B))]|_] for ...
```

Why Stop With Existential Conjectures?

```
tff(q,question,![A:$int,B:$int]:?[R:$int]:
        $difference(A,R)=B).

% SZS status Theorem for ...
% SZS answers Tuple
        [λA,B.[R->$uminus($sum($uminus(A),B))]|_] for ...
```

With answer literals, it is easy to also support the $\forall\exists$ questions

- we just need to Skolemize and remember the skolem names for a reverse replacement (a trick borrowed from `-qa synthesis`)

Why Stop With Existential Conjectures?

```
tff(q,question,![A:$int,B:$int]:?[R:$int]:
        $difference(A,R)=B).

% SZS status Theorem for ...
% SZS answers Tuple
        [λA,B.[R->$uminus($sum($uminus(A),B))]|_] for ...
```

With answer literals, it is easy to also support the ∀∃ questions

- we just need to Skolemize and remember the skolem names for a reverse replacement (a trick borrowed from `-qa synthesis`)

```
...
2. ~! [X0 : $int,X1 : $int] : ? [X2 : $int] :
      $difference(X0,X2) = X1 [negated conjecture 1]
3. ~? [X2 : $int] :
      (sK2_in = $difference(sK1_in,X2) & ans0(X2))
      [answer literal with input var skolemisation 2]
```

The Multiple Answers Conundrum

Already the "`SZS answers Tuple [[Answer]|_]`" suggests that there are potentially further answers to be found . . .

The Multiple Answers Conundrum

Already the "`SZS answers Tuple [[Answer]|_]`" suggests that there are potentially further answers to be found . . .

Thanks to Giles Reger's past efforts, we had an experimental branch with a support of returning more than one answer.

The Multiple Answers Conundrum

Already the "`SZS answers Tuple [[Answer]|_]`" suggests that there are potentially further answers to be found . . .

Thanks to Giles Reger's past efforts, we had an experimental branch with a support of returning more than one answer.

However, saturation-based search is a commitment to finding ONE proof (if it exists). Once one is found (along with ONE answer), doors to finding different ones might have already closed.

The Multiple Answers Conundrum

Already the "`SZS answers Tuple [[Answer]|_]`" suggests that there are potentially further answers to be found . . .

Thanks to Giles Reger's past efforts, we had an experimental branch with a support of returning more than one answer.

However, saturation-based search is a commitment to finding ONE proof (if it exists). Once one is found (along with ONE answer), doors to finding different ones might have already closed.

How else could we get to the other answers?

Multiple Answers via Repeated Invocations

Let the user specify what answers they have already seen.

Multiple Answers via Repeated Invocations

Let the user specify what answers they have already seen.

```
fof(a,axiom,?[X]:p(X)). fof(b,axiom,![X]:(p(X)=>p(a))).
fof(c,question,?[X]:p(X)).
```

Multiple Answers via Repeated Invocations

Let the user specify what answers they have already seen.

```
fof(a,axiom,?[X]:p(X)). fof(b,axiom,![X]:(p(X)=>p(a))).
fof(c,question,?[X]:p(X)).
```

```
% SZS answers Tuple [[X->sK1]|_] for fromPetra.p
%    sK1 introduced for X0 in 1. ?[X0]:p(X0) [input(axiom)]
% SZS output start Proof for fromPetra.p
...
21. ~ans0(sK1) [resolution 12,10]
23. ans0(X0) [answer literal resolver]
26. $false [unit resulting resolution 21,23]
```

Multiple Answers via Repeated Invocations

Let the user specify what answers they have already seen.

```
fof(a,axiom,?[X]:p(X)). fof(b,axiom,![X]:(p(X)=>p(a))).
fof(c,question,?[X]:p(X)).
```

```
% SZS answers Tuple [[X->sK1]|_] for fromPetra.p
%    sK1 introduced for X0 in 1. ?[X0]:p(X0) [input(axiom)]
% SZS output start Proof for fromPetra.p
...
21. ~ans0(sK1) [resolution 12,10]
23. ans0(X0) [answer literal resolver]
26. $false [unit resulting resolution 21,23]
```

```
./vampire -qa plain -qaat "ans0(sK1)" fromPetra.p
...
% SZS answers Tuple [[X->a]|_] for fromPetra.p
```

Goes Well with Vampire's New Interactive Mode

Load your large and expensive-to-parse knowledge base once via:

```
rlwrap ./vampire largeKB.p --interactive on
```

and enter an interactive session with Vampire.

Goes Well with Vampire's New Interactive Mode

Load your large and expensive-to-parse knowledge base once via:

```
rlwrap ./vampire largeKB.p --interactive on
```

and enter an interactive session with Vampire. State your question as in

```
tptp fof(q,question,?[Cnty,Ppl]:
            (hasPpl(Cnty,Ppl) & $less(1000000000,Ppl))).
```

Goes Well with Vampire's New Interactive Mode

Load your large and expensive-to-parse knowledge base once via:

```
rlwrap ./vampire largeKB.p --interactive on
```

and enter an interactive session with Vampire. State your question as in

```
tptp fof(q,question,?[Cnty,Ppl]:
            (hasPpl(Cnty,Ppl) & $less(1000000000,Ppl))).
```

Ask once:

```
run
% SZS answers Tuple [[Cnty->china,Ppl->1425490000]|_] for ...
```

Goes Well with Vampire's New Interactive Mode

Load your large and expensive-to-parse knowledge base once via:

```
rlwrap ./vampire largeKB.p --interactive on
```

and enter an interactive session with Vampire. State your question as in

```
tptp fof(q,question,?[Cnty,Ppl]:
            (hasPpl(Cnty,Ppl) & $less(1000000000,Ppl))).
```

Ask once:

```
run
% SZS answers Tuple [[Cnty->china,Ppl->1425490000]|_] for ...
```

Ask more:

```
run -qaat ans0(china,X)
% SZS answers Tuple [[Cnty->india,Ppl->1435230000]|_] for ...
```

Goes Well with Vampire's New Interactive Mode

Load your large and expensive-to-parse knowledge base once via:

```
rlwrap ./vampire largeKB.p --interactive on
```

and enter an interactive session with Vampire. State your question as in

```
tptp fof(q,question,?[Cnty,Ppl]:
            (hasPpl(Cnty,Ppl) & $less(1000000000,Ppl))).
```

Ask once:

```
run
% SZS answers Tuple [[Cnty->china,Ppl->1425490000]|_] for ...
```

Ask more:

```
run -qaat ans0(china,X)
% SZS answers Tuple [[Cnty->india,Ppl->1435230000]|_] for ...
```

Even more:

```
run -qaat ans0(china,X)|ans0(india,Y)
% Refutation not found, incomplete strategy
```

Goes Well with Vampire's New Interactive Mode

Load your large and expensive-to-parse knowledge base once via:

```
rlwrap ./vampire largeKB.p --interactive on
```

and enter an interactive session with Vampire. State your question as in

```
tptp fof(q,question,?[Cnty,Ppl]:
            (hasPpl(Cnty,Ppl) & $less(1000000000,Ppl))).
```

Ask once:

```
run
% SZS answers Tuple [[Cnty->china,Ppl->1425490000]|_] for ...
```

Ask more:

```
run -qaat ans0(china,X)
% SZS answers Tuple [[Cnty->india,Ppl->1435230000]|_] for ...
```

Even more:

```
run -qaat ans0(china,X)|ans0(india,Y)
% Refutation not found, incomplete strategy
```

```
^D
Bye. (And thank you, Michael, for suggesting rlwrap!)
```

Summary

A revised question answering mode in Vampire

- Based on the "standard" answer-literal trick
- AVATAR supported (except for Z3, as currently no cores)
- supports the synthesis-inspired $\forall\exists$ questions
- philosophy: affect proof search as little as possible
- variables in answers, Skolems in answers, disjunctive answers
- multiple answers? $\Rightarrow$ repeated invocation / explicit exclusion
- interactive mode could be useful too

Acknowledgments

- motivated by work on SUMO
- several improvements suggested by Adam Pease