Re: Exception to the rule? (Tarski´s T−scheme)

Source: http://sci.tech−archive.net/Archive/sci.logic/2004−06/1671.html

---

From: Paul Holbach (*paulholbachSPAMBAN_at_freenet.de*)
Date: 06/23/04

```
Date: 23 Jun 2004 07:53:18 −0700
```

> "Jeffrey Ketland" <ketland@ketland.fsnet.co.uk> wrote in message news:<
> cbafg3$5qo$1@news8.svr.pol.co.uk>...
> > Paul Holbach (paulholbachSPAMBAN@freenet.de) wrote in message
> > <881c8779.0406221452.42ac0931@posting.google.com>...

> >So we must write:
> >
> >[T("p") <−> p] −> Ex(x = x)

> You cannot compress the T−scheme to a single sentence, since it is a scheme,
> and thus has infinitely many instances.

Well, right.

> However, let DT be the theory in the
> language of arithmetic L_{Tr} extended with a primitive predicate Tr(x),
> containing the restricted T−scheme
>
> Tr("phi")<−>phi,
>
> for all sentences phi not containing Tr.

So "phi" cannot be eg "This sentence is not true".

> Then
>
> DT |− ExEy(x =/= y).
> (for proof, see below)

> >You are right−−iff the T−scheme in fact implies the existence of something.

> The T−scheme is actually inconsistent (with syntax), as Tarski pointed out
> in 1933.
> If you restrict it, so that in
> Tr("phi") <−> phi,
> the formula phi does not contain the symbol Tr, then it is easy to show that

> the scheme implies ExEy(x =/= y). That is, the restricted T−scheme implies
> the existence of at least two objects.
>
> Indeed, take any formula you like. Say 0=0.
> Then
>
> {Tr("0=0") <−> 0=0, Tr("~(0=0)") <−> ~(0=0)} |− ExEy(x =/= y)
>
> Proof:
>
> 1. Tr("0=0") <−> 0=0,
> 2. Tr("~(0=0)") <−> ~(0=0)
> 3. ~ExEy(x =/= y)
> |
> 4. AxAy(x = y)
> |
> 5. Tr("~(0=0)") <−> ~Tr("0=0")
> 6. "~(0=0)" = "0=0"
> 7. Tr("0=0") <−> ~Tr("0=0")
> contradiction

I see.
And the existence of at least two things certainly implies the
existence of something: Ex(x = x)

> >The restriction you´re referring to is Ex(x = x) / ExE!x, isn´t it?

> The restriction I mean is the restriction needed to avoid the Liar paradox,
> that in instances of Tr("phi") <−> phi, the formula phi does not contain the
> truth predicate symbol Tr.
>
> It is easy to show that the theory PA + T−scheme is inconsistent.
> Proof: By the Diagonal Lemma, construct a LIar sentence L such that PA |− L
> <−> ~Tr("L"). This contradicts the instance L<−> Tr("L"). So, PA +
> (unrestricted) T−scheme is inconsistent.

Paraconsistentists such as Graham Priest willingly bite the bullet,
accepting both the unrestricted T−scheme and inconsistency. For him "L
<−> ~L", i.e. "L & ~L" is true.

> Roughly, the restricted T−scheme is not self−applicative, and only applies
> to the formulas of the _object−language_, and intuitively the formula Tr(x)
> means "x is a true sentence of the object language".

I see.
Unless I´m prepared to sacrifice consistency on the altar of
dialetheism, I simply have to use the restricted T−scheme, haven´t I?

> *It can be shown that PA + restricted T−scheme is a conservative extension of*
> *PA. So, if you can prove an arithmetic formula in the theory PA + restricted*
> *T−scheme, then it is provable in PA. In fact, this holds for a very wide*
> *class of theories (certainly for any reasonable theory of syntax). In this*
> *sense, the (restricted) T−scheme represents a very weak notion of truth, and*
> *I have argued in print that this is why it can be considered "deflationary".*
> *(See my paper in "Deflationism and Tarski's Paradise", Mind 108 (1999), pp.*
> *69−94.)*
>
> *But you are correct in noting that the T−scheme (even the consistent,*
> *restricted T−scheme) has non−trivial ontological commitment: it implies the*
> *existence of at least two things (roughly, some sentence and its negation).*
> *So, the T−scheme implies "Something exists".*

Right.

[T("phi") <−> phi] −> Ex(x = x)

equivalently:

~Ex(x = x) −> ~[T("phi") <−> phi]

Since in classical FOPL "Ex(x = x)" is a logical, i.e. necessary
truth, "~Ex(x = x)" is impossibly true therein, whereas, for example,
in free logic "Ex(x = x)" is not a necessary but a contingent truth,
and so in free logic it could be the case that ~Ex(x = x), even though
it actually isn´t the case that ~Ex(x = x).

As I have already noticed, even if

1 e v("~Ex(x = x)") and 0 e v["T('~Ex(x = x)')"] , then

1 e v["~Ex(x = x) −> ~[T('~Ex(x = x)') <−> ~Ex(x = x)]"]

My initial point was that if it were the case that nothing exists,
then there would be neither truths nor falsities, that is, neither
true sentences nor false sentences.
In other words, if nothing existed, something would be the case but
nothing would be true!

> *However, also note that so long as your background theory is not*
> *ridiculously weak, the instance of the T−scheme with the sentence "Nothing*
> *exists", i.e., the formula*
>
> *Tr("~Ex(x =x)") <−> ~Ex(x = x)*
>
> *will be a perfectly acceptable theorem. It's just that both sides of the*
> *biconditional are false.*

OK, its equivalent is

F("~Ex(x = x)") <−> Ex(x = x) .

> *When people first come across the T−scheme, they often forget to consider*
> *that it also applies to instances where the sentence of the object language*
> *used is obviously untrue. For example,*
>
> *"The moon is made of cheese" is true iff the moon is made of cheese.*
> *"0 =1" is true iff 0 = 1*
> *etc.*

I´ve been aware of that.

T("p") <−> p |/− T("p")

T("p") <−> p |/− p

Regards
PH