

The Surprise Paradox

Joachim Breitner
Programming Paradigms Group
Karlsruhe Institute for Technology
breitner@kit.edu

23. Februar 2021

Zusammenfassung

In 1964, Fitch showed that the paradox of the surprise hanging can be resolved by showing that the judge's verdict is inconsistent. His formalization builds on Gödel's coding of provability.

In this theory, we reproduce his proof in Isabelle, building on Paulson's formalisation of Gödel's incompleteness theorems.

Inhaltsverzeichnis

1 Excluded or	2
2 Formulas with variables	2
3 Fitch's proof	4
4 Substitution, quoting and V-quoting	5

```
theory Surprise-Paradox  
imports  
  Incompleteness.Goedel-I  
  Incompleteness.Pseudo-Coding  
begin
```

The Surprise Paradox comes in a few variations, one being the following:

A judge sentences a felon to death by hanging, to be executed at noon the next week, Monday to Friday. As an extra punishment, the judge does not disclose the day of the hanging and promises the felon that it will come at a surprise.

The felon, probably a logician, then concludes that he cannot be hanged on Friday, as by then it would not longer be a surprise. Using this fact and similar reasoning, he cannot be hanged on

Thursday, and so on. He reaches the conclusion that he cannot be hanged at all, and contently returns to his cell.

Wednesday, at noon, the hangman comes to the very surprised felon, and executes him.

Obviously, something is wrong here: Does the felon reason wrongly? It looks about right. Or is the judge lying? But his prediction became true!

It is an interesting exercise to try to phrase the Surprise Paradox in a rigorous manner, and see this might clarify things.

In 1964, Frederic Fitch suggested a formulation that refines the notion of “surprise” as “cannot be proven from the given assumptions” [1]. To formulate that, we need propositions that reference their own provability, so just as Fitch builds on Gödel’s work, we build on Paulson’s formalisation of Gödel’s incompleteness theorems in Isabelle [2].

1 Excluded or

Although the proof goes through with regular disjunction, Fitch phrases the judge’s proposition using exclusive or, so we add syntax for that.

abbreviation $Xor :: fm \Rightarrow fm \Rightarrow fm$ (**infix** XOR 120)
where $Xor A B \equiv (A OR B) AND ((Neg A) OR (Neg B))$

2 Formulas with variables

In Paulson’s formalisation of terms and formulas, only terms carry variables. This is sufficient for his purposes, because the proposition that is being diagonalised needs itself as a parameter to $PfP::tm \Rightarrow fm$, which does take a term (which happens to be a quoted formula).

In order to stay close to Fitch, we need the diagonalised proposition to occur deeper in a quotation of a few logical conjunctions. Therefore, we build a small theory of formulas with variables (“holed” formulas). These support substituting a formula for a variable, this substitution commutes with quotation, and closed holed formulas can be converted to regular formulas.

In our application, we do not need holes under an quantifier, which greatly simplifies things here. In particular, we can use **datatype** and **fun**.

datatype $hfm =$
 $HVar$ name
 | HFm fm
 | $HDisj$ hfm hfm (**infixr** HOR 130)
 | $HNeg$ hfm

abbreviation $HImp :: hfm \Rightarrow hfm \Rightarrow hfm$ (**infixr** HIMP 125)

where $HImp\ A\ B \equiv HDisj\ (HNeg\ A)\ B$

definition $HConj :: hfm \Rightarrow hfm \Rightarrow hfm$ (**infixr** *HAND* 135)
where $HConj\ A\ B \equiv HNeg\ (HDisj\ (HNeg\ A)\ (HNeg\ B))$

abbreviation $HXor :: hfm \Rightarrow hfm \Rightarrow hfm$ (**infix** *HXOR* 120)
where $HXor\ A\ B \equiv (A\ HOR\ B)\ HAND\ (HNeg\ A\ HOR\ HNeg\ B)$

fun $subst\text{-}hfm :: hfm \Rightarrow name \Rightarrow fm \Rightarrow hfm$ ($\text{'(-::=-)'} [1000, 0, 0] 200$)
where
 $(HVar\ name)(i::=x) = (if\ i = name\ then\ HFm\ x\ else\ HVar\ name)$
 $| (HDisj\ A\ B)(i::=x) = HDisj\ (A(i::=x))\ (B(i::=x))$
 $| (HNeg\ A)(i::=x) = HNeg\ (A(i::=x))$
 $| (HFm\ A)(i::=x) = HFm\ A$

lemma $subst\text{-}hfm\text{-}Conj[simp]$:
 $(HConj\ A\ B)(i::=x) = HConj\ (A(i::=x))\ (B(i::=x))$
unfolding $HConj\text{-}def$ **by** $simp$

instantiation $hfm :: quot$
begin

fun $quot\text{-}hfm :: hfm \Rightarrow tm$
where
 $quot\text{-}hfm\ (HVar\ name) = (Var\ name)$
 $| quot\text{-}hfm\ (HFm\ A) = \langle A \rangle$
 $| quot\text{-}hfm\ (HDisj\ A\ B) = HPair\ (HTuple\ 3)\ (HPair\ (quot\text{-}hfm\ A)\ (quot\text{-}hfm\ B))$
 $| quot\text{-}hfm\ (HNeg\ A) = HPair\ (HTuple\ 4)\ (quot\text{-}hfm\ A)$

instance ..
end

lemma $subst\text{-}quot\text{-}hfm[simp]$: $subst\ i\ \langle P \rangle\ \langle A \rangle = \langle A(i::=P) \rangle$
by ($induction\ A$) $auto$

fun $hfm\text{-}to\text{-}fm :: hfm \Rightarrow fm$
where
 $hfm\text{-}to\text{-}fm\ (HVar\ name) = undefined$
 $| hfm\text{-}to\text{-}fm\ (HFm\ A) = A$
 $| hfm\text{-}to\text{-}fm\ (HDisj\ A\ B) = Disj\ (hfm\text{-}to\text{-}fm\ A)\ (hfm\text{-}to\text{-}fm\ B)$
 $| hfm\text{-}to\text{-}fm\ (HNeg\ A) = Neg\ (hfm\text{-}to\text{-}fm\ A)$

lemma $hfm\text{-}to\text{-}fm\text{-}Conj[simp]$:
 $hfm\text{-}to\text{-}fm\ (HConj\ A\ B) = Conj\ (hfm\text{-}to\text{-}fm\ A)\ (hfm\text{-}to\text{-}fm\ B)$
unfolding $HConj\text{-}def$ $Conj\text{-}def$ **by** $simp$

fun $closed\text{-}hfm :: hfm \Rightarrow bool$
where
 $closed\text{-}hfm\ (HVar\ name) \longleftrightarrow False$
 $| closed\text{-}hfm\ (HFm\ A) \longleftrightarrow True$

| $closed\text{-}hfm (HDisj A B) \longleftrightarrow closed\text{-}hfm A \wedge closed\text{-}hfm B$
| $closed\text{-}hfm (HNeg A) \longleftrightarrow closed\text{-}hfm A$

lemma $closed\text{-}hfm\text{-}Conj[simp]$:
 $closed\text{-}hfm (HConj A B) \longleftrightarrow closed\text{-}hfm A \wedge closed\text{-}hfm B$
unfolding $HConj\text{-}def$ **by** $simp$

lemma $quot\text{-}closed\text{-}hfm[simp]$: $closed\text{-}hfm A \implies \langle\langle A \rangle\rangle = \langle\langle hfm\text{-}to\text{-}fm A \rangle\rangle$
by ($induction A$) ($auto simp add: quot\text{-}fm\text{-}def$)

declare $quot\text{-}hfm.simps[simp del]$

3 Fitch's proof

For simplicity, Fitch (and we) restrict the week to two days. Propositions Q_1 and Q_2 represent the propositions that the hanging occurs on the first resp. the second day, but these can obviously be any propositions.

context
fixes $Q_1 :: fm$ **and** $Q_2 :: fm$
assumes $Q\text{-closed}: supp Q_1 = \{\} supp Q_2 = \{\}$
begin

In order to define the judge's proposition, which is self-referential, we apply the usual trick of defining a proposition with a variable, and then using Gödel's diagonalisation lemma.

definition $H :: fm$ **where**
 $H = Q_1 AND Neg (PfP \langle\langle HVar X0 HIMP HFm Q_1 \rangle\rangle) XOR$
 $Q_2 AND Neg (PfP \langle\langle HVar X0 HAND HNeg (HFm Q_1) HIMP (HFm Q_2) \rangle\rangle)$

definition P **where** $P = (SOME P. \{\} \vdash P IFF H(X0 ::= \langle\langle P \rangle\rangle))$

lemma P' : $\{\} \vdash P IFF H(X0 ::= \langle\langle P \rangle\rangle)$

proof–
from $diagonal[\mathbf{where} \alpha = H \mathbf{and} i = X0]$
obtain δ **where** $\{\} \vdash \delta IFF H(X0 ::= \langle\langle \delta \rangle\rangle)$.
thus $?thesis$ **unfolding** $P\text{-def}$ **by** ($rule someI$)
qed

From now on, the lemmas are named after their number in Fitch's paper, and correspond to his statements pleasingly closely.

lemma γ : $\{\} \vdash P IFF$
 $(Q_1 AND Neg (PfP \langle\langle P IMP Q_1 \rangle\rangle) XOR$
 $Q_2 AND Neg (PfP \langle\langle P AND Neg Q_1 IMP Q_2 \rangle\rangle))$
using P' **unfolding** $H\text{-def}$
by ($simp add: Q\text{-closed forget\text{-}subst\text{-}fm[unfolding fresh\text{-}def]$)
lemmas $\gamma\text{-E} = \gamma[THEN thin0, THEN Iff\text{-}MP\text{-}left', OF Conj\text{-}E, OF thin2]$

lemmas *propositional-calculus* =
AssumeH Neg-I Imp-I Conj-E Disj-E ExFalso[OF Neg-E]
ExFalso[OF rotate2, OF Neg-E] ExFalso[OF rotate3, OF Neg-E]

lemma 8: $\{\} \vdash (P \text{ AND } \text{Neg } Q_1) \text{ IMP } Q_2$
by (*intro propositional-calculus 7-E*)

lemma 10: $\{\} \vdash \text{PfP} \langle\langle P \text{ AND } \text{Neg } Q_1 \rangle\rangle \text{ IMP } Q_2$
using 8 by (*rule proved-imp-proved-PfP*)
lemmas 10-I = *10[THEN thin0]*

lemma 11: $\{\} \vdash P \text{ IMP } Q_1$
by (*intro propositional-calculus 7-E 10-I*)

lemma 12: $\{\} \vdash \text{PfP} \langle\langle P \text{ IMP } Q_1 \rangle\rangle$
using 11 by (*rule proved-imp-proved-PfP*)
lemmas 12-I = *12[THEN thin0]*

lemma 13: $\{\} \vdash \text{Neg } P$
by (*intro propositional-calculus 7-E 10-I 12-I*)

end

4 Substitution, quoting and V-quoting

In the end, we did not need the lemma at the end of this section, but it may be useful to others.

lemma *trans-tm-forgets:* *atom ' set is #* t \implies trans-tm is t = trans-tm [] t*
by (*induct t rule: tm.induct*)
(auto simp: lookup-notin fresh-star-def fresh-at-base)

lemma *vquot-dbtm-fresh:* *atom ' V #* t \implies vquot-dbtm V t = quot-dbtm t*
by (*nominal-induct t rule: dbtm.strong-induct*)
(auto simp add: fresh-star-def fresh-at-base)

lemma *subst-vquot-dbtm-trans-tm[simp]:*
atom i # is \implies atom ' set is # t \implies*
subst i «t» (vquot-dbtm {i} (trans-tm is t')) =
quot-dbtm (trans-tm is (subst i t t'))
by (*nominal-induct t' avoiding: is i t rule: tm.strong-induct*)
(auto simp add: quot-tm-def lookup-notin fresh-imp-notin-env
vquot-dbtm-fresh lookup-fresh
intro: trans-tm-forgets[symmetric])

lemma *subst-vquot-dbtm-trans-fm[simp]:*
atom i # is \implies atom ' set is # t \implies*
subst i «t» (vquot-dbfm {i} (trans-fm is A)) =
quot-dbfm (trans-fm is (subst-fm A i t))
by (*nominal-induct A avoiding: is i t rule: fm.strong-induct*)

(*auto simp add: quot-fm-def fresh-Cons*)

lemma *subst-vquot[simp]*:

subst i «t» [A]{i} = «A(i ::= t)»

by (*nominal-induct A avoiding: i t rule: fm.strong-induct*)

(*auto simp add: vquot-fm-def quot-fm-def fresh-Cons*)

end

Literatur

- [1] F. B. Fitch. A goedelized formulation of the prediction paradox. *American Philosophical Quarterly*, 1(2):161–164, 1964.
- [2] L. C. Paulson. Gödel’s incompleteness theorems. *Archive of Formal Proofs*, Nov. 2013. <http://isa-afp.org/entries/Incompleteness.shtml>, Formal proof development.