

MAT 7003 : Mathematical Foundations

(for Software Engineering)

J Paul Gibson, A207

paul.gibson@it-sudparis.eu

<http://www-public.it-sudparis.eu/~gibson/Teaching/MAT7003/>

Logic – Sample Answers

[.../~gibson/Teaching/MAT7003/L4-Logic-SampleAnswers.pdf](http://www-public.it-sudparis.eu/~gibson/Teaching/MAT7003/L4-Logic-SampleAnswers.pdf)

How to Use Propositional Logic?

Problem: Determine the validity of the following reasoning

« Paul can eat if he has cooked a meal in the last 30 minutes and he is hungry. He is hungry provided he has not eaten in the last 15 minutes and in the last 30 minutes he has not done any sport.

If he is doing sport he cannot cook a meal at the same time. Therefore if Paul did not eat he had already eaten in the last 15 minutes or he did sport in the last 30 minutes »

Formalising the reasoning using the set BOOL ({TRUE, FALSE}) in Event-B

CONTEXT

```

WhyIsPaulNotEating // Determine the validity of the following reasoning
// « Paul can eat if he has cooked a meal in the last 30 minutes
// and he is hungry. He is hungry provided he has not eaten in the
// last 15 minutes and in the last 30 minutes he has not done any sport.
// If he is doing sport he cannot cook a meal at the same
// time. Therefore if Paul did not eat he had already eaten
// in the last 15 minutes or he did sport in the last 30 minutes »

```

CONSTANTS

```

P_Eating // Paul is eating
P_Cooked // Paul has cooked a meal in the last 30 minutes
P_Hungry // Paul is hungry
P_Eaten // Paul has eaten in last 15 minutes
P_Sport // Paul has done sport in last 30 minutes

```

AXIOMS

```

axm1 : P_Eating ∈ BOOL // Paul is eating
axm2 : P_Cooked ∈ BOOL // Paul has cooked a meal in the last 30 minutes
axm3 : P_Hungry ∈ BOOL // Paul is hungry
axm4 : P_Eaten ∈ BOOL // Paul has eaten in last 15 minutes
axm5 : P_Sport ∈ BOOL // Paul has done sport in last 30 minutes

axm6 : (P_Eating = TRUE) ⇒ // Paul can eat is he has cooked a meal in the last
((P_Cooked = TRUE) ∧ // 30 minutes and he is hungry
(P_Hungry = TRUE))

axm7 : (P_Hungry = TRUE) ⇒ // Paul is hungry if he has not eaten in the last
((P_Eaten = FALSE) ∧ // 15 minutes and he has not done sport in the
(P_Sport = FALSE)) // last 30 minutes

axm8 : ¬ ((P_Sport = TRUE) ∧ // If he is doing sport he cannot cook a meal
(P_Cooked = TRUE)) // at the same time

axm9 : ¬ ((P_Eating = TRUE) ⇒ // If Paul did not eat he had already eaten
((P_Eaten = TRUE) ∨ // in the last 15 minutes or he did sport
(P_Sport = TRUE)) // in the last 30 minutes

axm10 : ¬ ((¬ (P_Eating = TRUE) ⇒ // not (if Paul did not eat he had already eaten
((P_Eaten = TRUE) ∨ // in the last 15 minutes or he did sport
(P_Sport = TRUE)) // in the last 30 minutes)

```

END

NOTE: The importance of the precise interpretation of the natural language text; in particular words such as: **can**, should, could, must, may.

Intuitively, the reasoning (expressed in theorem **axm9**) must be false as nothing obliges Paul to eat (or cook).

We will see this when we launch the Rodin prover on this theorem ...

Formalising the reasoning using the set BOOL ({TRUE, FALSE}) in Event-B

The screenshot shows the Rodin prover interface. On the left, a proof tree is displayed, showing a goal that is not proven. On the right, a counterexample state is shown, indicating that the theorem is false. The state is:

- P_Eating=FALSE
- P_Eaten=FALSE
- P_Sport=FALSE
- ¬ P_Cooked=TRUE

The goal is 1.

The prover shows that the theorem is false when Paul is not eating, he has not eaten, he has not done sport and has not cooked

Alternatively, we can also use the prover directly to show that the reasoning is wrong

```

CONTEXT
    // Determine the validity of the following reasoning
    // « Paul can eat if he has cooked a meal in the last 30 minutes
    // and he is hungry. He is hungry provided he has not eaten in the
    // last 15 minutes and in the last 30 minutes he has not done any sport.
    // If he is doing sport he cannot cook a meal at the same
    // time. Therefore, if Paul did not eat he had already eaten
    // in the last 15 minutes or he did sport in the last 30 minutes »

WhyIsPaulNotEating

CONSTANTS
P_Eating // Paul is eating
P_Cooked // Paul has cooked a meal in the last 30 minutes
P_Hungry // Paul is hungry
P_Eaten // Paul has eaten in last 15 minutes
P_Sport // Paul has done sport in last 30 minutes

AXIOMS
axm1 : P_Eating ∈ BOOL // Paul is eating
axm2 : P_Cooked ∈ BOOL // Paul has cooked a meal in the last 30 minutes
axm3 : P_Hungry ∈ BOOL // Paul is hungry
axm4 : P_Eaten ∈ BOOL // Paul has eaten in last 15 minutes
axm5 : P_Sport ∈ BOOL // Paul has done sport in last 30 minutes

axm6 : (P_Eating = TRUE) ⇒ // Paul can eat is he has cooked a meal in the last
((P_Cooked = TRUE) ∧ // 30 minutes and he is hungry
(P_Hungry = TRUE))

axm7 : (P_Hungry = TRUE) ⇒ // Paul is hungry if he has not eaten in the last
((P_Eaten = FALSE ∧ // 15 minutes and he has not done sport in the
P_Sport = FALSE)) // last 30 minutes

axm8 : ¬ ((P_Sport = TRUE) ∧ // If he is doing sport he cannot cook a meal
(P_Cooked = TRUE)) // at the same time

axm9 : P_Cooked = FALSE ∧ P_Eaten = FALSE ∧ P_Sport = FALSE

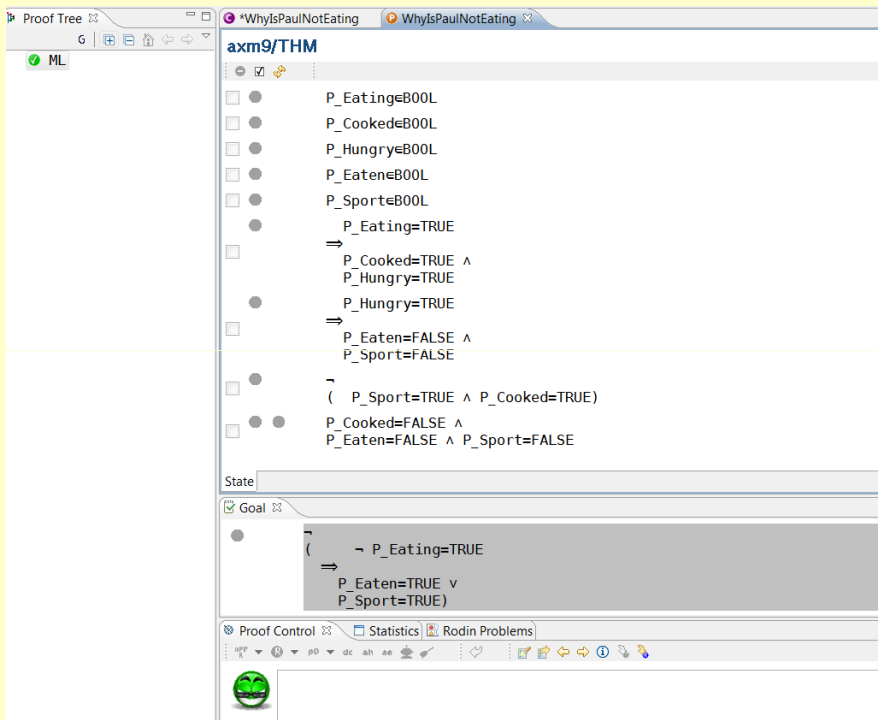
axm10 : ¬ (¬ (P_Eating = TRUE) ⇒ // not (Therefore, if Paul did not eat he had already eaten
((P_Eaten = TRUE) ∨ // in the last 15 minutes or he did sport
(P_Sport = TRUE)) // in the last 30 minutes)

END
    
```

We now know under what conditions the reasoning fails

Theorem: the reasoning is wrong

Alternatively, we can also use the prover directly to show that the reasoning is wrong



TO DO:
Change the specification so that Paul is obliged to cook and eat when he is hungry

Specifications in Predicate Logic

Which of the following formulae are valid (true), and why?

- 1 lemma " $(\exists x. \forall y. P x y) \longrightarrow (\forall y. \exists x. P x y)$ "
- 2 lemma " $(\forall x. P x \longrightarrow Q) = ((\exists x. P x) \longrightarrow Q)$ "
- 3 lemma " $((\forall x. P x) \wedge (\forall x. Q x)) = (\forall x. (P x \wedge Q x))$ "
- 4 lemma " $((\forall x. P x) \vee (\forall x. Q x)) = (\forall x. (P x \vee Q x))$ "
- 5 lemma " $((\exists x. P x) \vee (\exists x. Q x)) = (\exists x. (P x \vee Q x))$ "
- 6 lemma " $(\forall x. \exists y. P x y) \longrightarrow (\exists y. \forall x. P x y)$ "
- 7 lemma " $(\neg (\forall x. P x)) = (\exists x. \neg P x)$ "

Specifications in Predicate Logic: lemma 7

This lemma is, perhaps, the most obviously true. So let's put it into RODIN in Event-B, using the BOOL set.

CONTEXT

SomeFormulae

SETS

Universe

CONSTANTS

PFun

AXIOMS

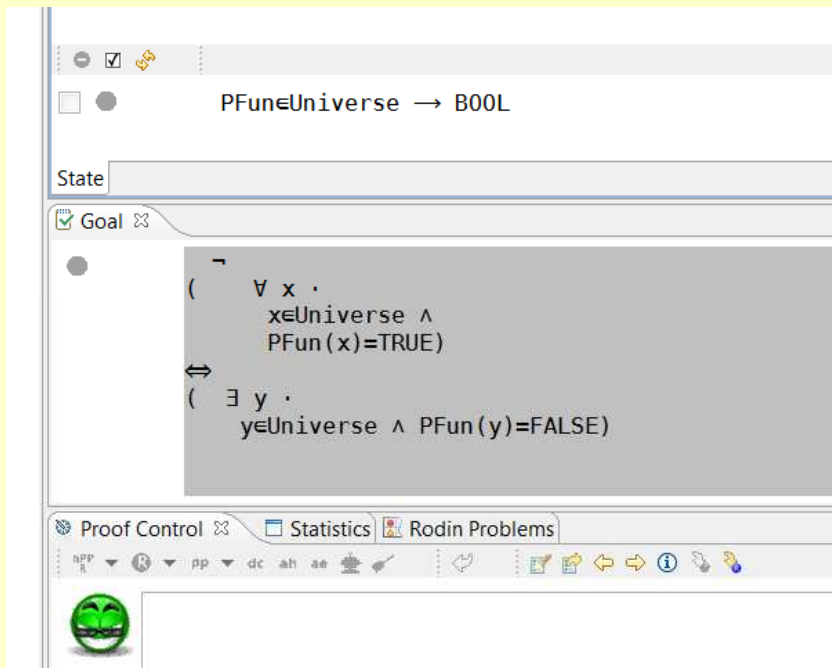
axml : PFun \in Universe \rightarrow BOOL

lemma7 : $\neg (\forall x. x \in \text{Universe} \wedge \text{PFun}(x) = \text{TRUE})$
 \Leftrightarrow
 $(\exists y. y \in \text{Universe} \wedge \text{PFun}(y) = \text{FALSE})$

END

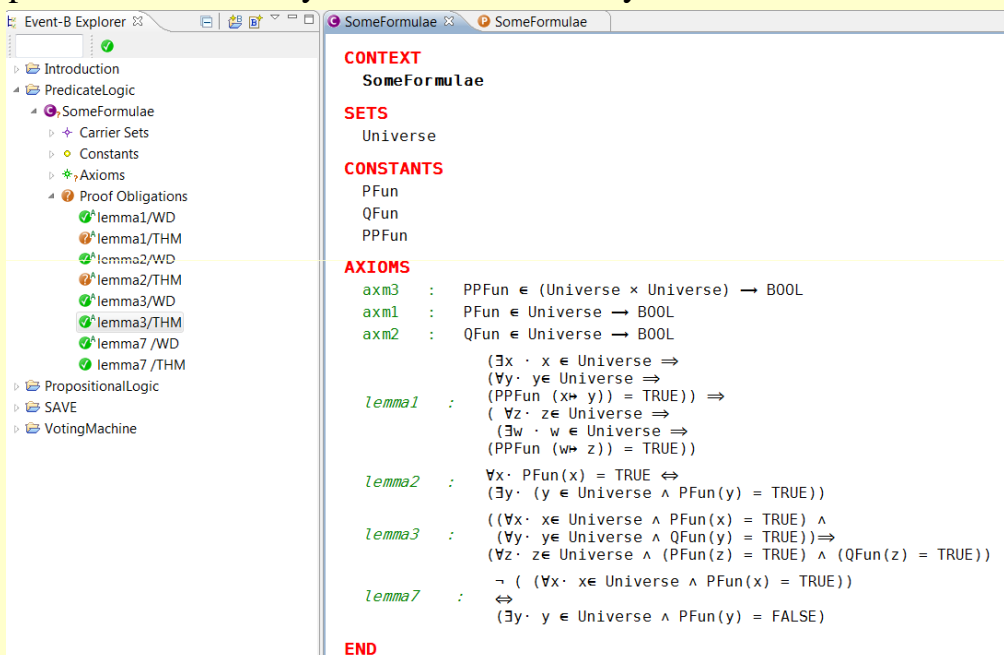
Specifications in Predicate Logic: lemma 7

Not surprisingly, the Prover discharges this automatically



Specifications in Predicate Logic: lemmas 1 to 3

The prover discharges 3 automatically, so we know it is true (provided the lemma is properly specified in Event-B). However, lemma1 and lemma2 are not proven automatically -does this mean they are false?



TO DO:
verify and complete the lemmas using the same style (based on BOOL functions)

Specifications in Predicate Logic: another more direct specification style

We could also represent the lemma in a more direct way using a set – PSet, say - to explicitly represent the subset of the universe for which the predicate P is true:

CONSTANTS

PSet

AXIOMS

```
axm4 : PSet ⊆ Universe
lemma7 : ¬ ( (∀x. x ∈ Universe ∧ PFun(x) = TRUE)
            ⇔
            (∃y. y ∈ Universe ∧ PFun(y) = FALSE) )
lemma7b : ¬ ( (∀x. x ∈ Universe ∧ x ∈ PSet)
              ⇔
              (∃y. y ∈ Universe ∧ y ∉ PSet ) )
```

QUESTION: Which of the representations is *best*?

TO DO: verify and complete the lemmas using this new style of specification (based on set membership)

Specifications in Predicate Logic

If a student₁ is good-at-logic, he₁ does-well-in-semantic.

QUESTION: Consider and judge the following possibilities:

- i. $[\exists x(ST(x) \wedge LOG(x))] \rightarrow SEM(x)$
- ii. $\exists x [(ST(x) \wedge LOG(x)) \rightarrow SEM(x)]$
- iii. $\forall x [(ST(x) \wedge LOG(x)) \rightarrow SEM(x)]$
- iv. $\exists x [ST(x) \wedge (LOG(x) \rightarrow SEM(x))]$

Specifications in Predicate Logic

Express the following using predicate logic

1. *Susan introduced Mary to a student that nobody liked.*
2. *Only-if John has-talked-to every witness will Mary be-satisfied.*
3. *John introduced only MARY to Kate.*
4. *John introduces Mary only to KATE.*

Specifications in Predicate Logic:

Susan introduced Mary to a student that nobody liked.

CONTEXT

Party

SETS

PEOPLE

CONSTANTS

Susan

Mary

John

Kate

introduced

student

liked

talkedTo

witness

satisfied

AXIOMS

axm1 : Susan ∈ PEOPLE

axm2 : Mary ∈ PEOPLE

axm3 : John ∈ PEOPLE

axm4 : Kate ∈ PEOPLE

axm5 : introduced ⊆ PEOPLE × PEOPLE × PEOPLE

axm6 : student ⊆ PEOPLE

axm7 : liked ⊆ PEOPLE × PEOPLE

axm8 : talkedTo ⊆ PEOPLE × PEOPLE

axm9 : witness ⊆ PEOPLE

axm10 : satisfied ⊆ PEOPLE

Exp1 : $\exists s. s \in \text{ran}(\text{liked}) \wedge$ // *Susan introduced Mary to a student that nobody liked.*
Susan ≠ Mary ∧ s ∈ introduced

END

TO DO: Complete
the context
specification

Specifications in Predicate Logic

Express the following using predicate logic

1. *All students in the class can program in Java or C++*
2. *Some of the students can program in Java or C++*
3. *Every student who could not program in Java or C++ was paired with a student who could*
4. *Some students program in Java some of the time, others never program in Java*

TO DO: Write an Event-B context to specify these properties