Politecnico di Milano - University of Illinois at Chicago
Formal Methods exam
Prof. Mandrioli

Proving TRIO theorems in PVS

By Daniele Paolo Scarpazza

1 Foreword

In this paper I will address the problem of trying to formally prove the TRIO theorems within the PVS environment. Please do not be scared by the length of this paper (approximately 60 pages), since only a few of the starting pages contain the human-written introduction to the proofs, with some reasoning and the justification for some choices I made.

Most of the paper length is constituted by the full log of the proofs themselves, it should be used only as a reference to check the consistency and the correctness of the very details of my work.

2 TRIO theorems

The theorems I will try to prove are taken from *Felder, Mandrioli, Morzenti*, "Proving properties of real-time systems through logical specifications and Petri net models".

During the writing of this text and the required work, I noticed that the overall proving work can be divided in the following three phases:

1 **rewriting**: in this phase the TRIO theorems are taken from the "pretty-printed" paper format and rewrote, by substituting symbols with the suitable PVS keywords; this is usually a mechanical phase and has been therefore successfully completed for all theorems.

2 **type correctness**: in this phase TRIO theorems are a little modified a in order to achieve type correctness according to how TRIO predicates are defines inside PVS. This phase is required since usually a boolean value is required as a theorem by PVS, whereas TRIO theorems are usually composed by time-dependent formulas that are implicitly referred to the current instant or, (in TRIO+) are automatically enclosed in an *Alw(.)* predicate. In this phase I usually tried both strategies, that is to say, I instantiated the whole theorem or the only time-dependent part at the current instant by using the *NOW(.)* predicate, or I stated the theorem in every instant, by enclosing it in an *Alw(.)*, therefore rendering it a time-independent boolean expression. Obviously by applying these strategies in different ways, a TRIO theorem can be rewritten in more TRIO-PVS theorems, and this is the reason why there are "theorem_?_bis", "theorem_?_ter" and so on. This phase too has been successfully completed for all the theorems.

3 **the very proof**: in this phase I tried to proof the theorems by applying suitable PVS strategies. Many of the theorems (a few less than a half) are simply proven by applying the *(grind)* rule, and for some of them, as an exercise, I decided to try a more "manual" proof, without using *(grind)* and applying more basic rules, such as *(skolem!), (expand "...")*, *(flatten), (instantiate ... "...")* and so on. Several theorems are proven with **my original contributions and efforts**, that is to say I had to invent and prove suitable lemmas, use them, find suitable strategies, since the conventional ones (such as *(grind)*) were not effective. Because of the inherent difficulties of the proving process, not all the theorems have been here successfully proven.

It was reported to me (and according to my limited experience it is true) that it is always possible to prove a theorem (**even a false one!!!**) by the use of the *(open-skolem ...)* rule. I decide not to use it at all, even if someone did use it, since it appeared to me as a fraudulent behaviour which brings to counterfeit proofs. I preferred to report instead also the unfinished and failed proofs, together with some comments and the reasoning on the difficulties I encountered, on the stratefies I tried to put in place in order to overcome them and so on...

3 Some little mistakes

While trying to prove the second and the third of the three assertions composing theorem vi, (page 11 of the original paper) I encountered difficulties whose strength were unjustified with respect to the simplicity of the theorems. Since only the first of the three theorems has been formally proven in the original paper, I thought that the other two, which I could not prove, were, in some sense, false.

I examined them more deeply and, even at an informal level, it was clear that they contained some sort of typo or mistake. In fact, trying to prove them I reached a point at which it was clear that the antecedents were not in accordance with the consequents; moreover, if I tried to modify the theorems in order to comply with what my intuit suggested me about the semantics of them, they were easily proven even by applying the *(grind)* rule.

Take a look at the following sections, subsections **theorem vi.b.** and **theorem vi.c.** for the details.

4 TRIO theorems' PVS transcription

4.1 theorem i.

Informal semantics: when a formula is always true, it is also true in all the instants who are at a distant of t time from the current one.

This theorem is rather simple to prove. Here are some possible TRIO transciptions for it:

Alw(α) $\Rightarrow$ Dist(α,t)

```
theorem_i: theorem
  Alw(Alw(a) IMPLIES Dist(a,t))

 theorem_i_bis: theorem
  Alw(a) IMPLIES NOW(Dist(a,t))

 theorem_i_ter: theorem
  NOW(Alw(a) IMPLIES Dist(a,t))
```

4.2 theorem ii.

Informal semantics: when a formula is always true, it can be derived that is true also in the current instant.

This theorem has been rather simple to prove.
Here are some possible TRIO transcriptions for it:

$\mathrm{Alw}(\alpha) \vdash \alpha$

```
theorem_ii: theorem
 Alw(Alw(a) IMPLIES a)

theorem_ii_bis: theorem
 Alw(a) IMPLIES NOW(a)

theorem_ii_ter: theorem
 NOW(Alw(a) IMPLIES a)
```

4.3 theorem iii.

Informal semantics: if a formula ϕ implies another formula ψ at the same time, and ϕ holds in an instant which is distant t from the current one, then the ψ formula too will hold in that instant.

$(\mathrm{Alw}(\phi \Rightarrow \psi) \wedge \mathrm{Dist}(\phi,t)\ ) \Rightarrow \mathrm{Dist}(\psi,t)$

This theorem is rather simple to prove.

These are two possible transcriptions for the theorem:

```
theorem_iii: theorem
  Alw((Alw(a IMPLIES b) AND Dist(a,t)) IMPLIES Dist(b,t))

theorem_iii_bis: theorem
  Alw(a IMPLIES b) AND NOW(Dist(a,t)) IMPLIES NOW(Dist(b,t))
```

4.4 theorem iv.

Informal semantics: If a time-independent formula is true at some instant different from the current one, then it is true now too.

$\mathrm{Dist}(\phi,x) \Rightarrow \phi$, where ϕ is a time-independent formula

This is my very personal transcription for this formula. In order to represent the fact that the formula is time-independent, I introduced the two hypotheses `( a IMPLIES Alw(a) ) AND ( (NOT a) IMPLIES (Alw(NOT a))`.

This theorem has been proven with a little efforts and difficulties, a simple *(grind)* did not manage to prove it. Please take a look at its proof details.

```
theorem_iv_complex: THEOREM
   Alw( (    ( a IMPLIES Alw(a) )
         AND ( (NOT a) IMPLIES (Alw(NOT a)))
         AND ( Dist(a,t))) IMPLIES a)
```

4.5 theorem v.

Informal semantics: the implication sign can be brought out of the *Futr(.,.)* TRIO operator.

Futr($\alpha \Rightarrow \beta$,t) $\Rightarrow$ (Futr(α,t) $\Rightarrow$ Futr(β,t))

Here are two possible transcription of the theorem.

```
theorem_v: THEOREM
 Alw( Futr(a IMPLIES b, t) IMPLIES (Futr(a,t) IMPLIES Futr(b,t)) )

theorem_v_bis: THEOREM
 NOW(Futr(a IMPLIES b, t) IMPLIES (Futr(a,t) IMPLIES Futr(b,t)) )
```

Theorem v is easily proven. A simple *(grind)*-based proof is reported in the next section.

4.6 theorem vi.

Informal semantics: these are the rules used to merge nested *Past* and *Futr* TRIO operators together, obtaining a formula which contains one TRIO temporal operator less than the original one.

$\text{Futr}(\text{Futr}(\alpha, t_1), t_2) \Rightarrow \text{Futr}(\alpha, t_1 + t_2)$

```
theorem_vi_a: THEOREM
 NOW(Futr(Futr(a,t1),t2) => Futr(a, t1 + t2))
```

As I said before, the following theorem, exactly as reported from the original paper, contains a mistake (the last two occurrences of the two instants t_1 and t_2 should be swap). Both versions (the original and the corrected one) of the theorem have been here reported. The first has been subject to a proof trial (failed) and the second one has a complete and successful proof.

$\text{Futr}(\text{Past}(\alpha, t_1), t_2) \wedge t_1 > t_2 \Rightarrow \text{Futr}(\alpha, t_1 - t_2)$

```
theorem_vi_b_original: THEOREM
 NOW(Futr(Past(a,t1),t2) AND t1 > t2 IMPLIES Futr(a, t1 - t2))

theorem_vi_b_correct: THEOREM
 NOW(Futr(Past(a,t1),t2) AND t1 > t2 IMPLIES Futr(a, t2 - t1))
```

The following theorem too, as already said, contains the same mistake in the original version as the previous one. Both versions of the theorem have been reported, the first has been subject to a failed proof trial and the second one has a complete and successful proof.

$\text{Futr}(\text{Past}(\alpha, t_1), t_2) \wedge t_1 < t_2 \Rightarrow \text{Past}(\alpha, t_2 - t_1)$

```
theorem_vi_c_original: THEOREM
 NOW(Futr(Past(a,t1),t2) AND t1 < t2 IMPLIES Past(a, t2 - t1))

theorem_vi_c_correct: THEOREM
 NOW(Futr(Past(a,t1),t2) AND t1 < t2 IMPLIES Past(a, t1 - t2))
```

4.7 theorem vii.

Informal semantics: the two parts which compose this theorem state that it is possible to bring the existential and the universal quantification out of the TRIO *Dist(.,.)* operator.

$\text{Dist}(\forall x\ \alpha, t) \Leftrightarrow \forall x \text{Dist}(\alpha, t)$

```
theorem_vii_a: THEOREM
 NOW(Dist(FA! (x): a,t) IMPLIES FA!(x): Dist(a,t))

theorem_vii_a_rev: THEOREM
 NOW(FA!(x): Dist(a,t) IMPLIES  Dist(FA! (x): a,t))
```

$\text{Dist}(\exists x\ \alpha, t) \Leftrightarrow \exists x \text{Dist}(\alpha, t)$

```
theorem_vii_b: THEOREM
 NOW(Dist(EX! (x): a,t) => EX!(x): Dist(a,t))

theorem_vii_b_rev: THEOREM
 NOW( EX!(x): Dist(a,t) => Dist(EX! (x): a,t))
```

All transcriptions of the theorem in both senses (there are double implications) can be easily proved. There's a log of their proof in the next section.

4.8 theorem viii.

Informal semantics: When a formula is always true, than it is always possible to bring it in instants different from the current one and still obtain a true formula.

$\text{Alw}(\alpha) \Leftrightarrow \text{Alw}(\text{Dist}(\alpha, s)$

The following transcription in PVS splits the double implication into two separate theorems. The first has a straightforward proof, the second requires a lemma which is given here.

```
theorem_viii: THEOREM
 Alw(a) => Alw(Dist(a,t))
```

```
inverse_add_time : THEOREM
 -t1 + t1 = 0

theorem_viii_rev: THEOREM
 Alw(Dist(a,t)) => Alw(a)
```

Warning: the reverse part of theorem viii was a not-so-easy-to-prove theorem (obviously the *(grind)* rule is not effective on it). I had to introduce a dedicated lemma, called "inverse_add_time", prove it first, and then use it to prove the theorem. Please take a look at the complete log of the proof, reported in the next section, since it contains an original contribution of mine.

4.9 theorem ix.

Informal semantics: ix states that if a system is periodic and the repeating property takes place at a given instant (the current instant, for simplicity), then the property will actually repeat itself indefinitely in all future times, at a regular pace.

$$\phi \wedge \mathrm{Alw}(\phi \ \Rightarrow \mathrm{Futr}(\phi, a)) \ \Rightarrow \ \forall k \mathrm{Futr}(\phi, a\ k)$$

Here are two sensible PVS transcriptions of the property:

```
theorem_ix: THEOREM
  NOW(a AND Alw(a IMPLIES Futr(a,p)) IMPLIES (FA! (k): Futr(a,k*p)))

theorem_ix_bis: THEOREM
  Alw( (a AND Alw(a IMPLIES Futr(a,t))) IMPLIES FA! (k): Futr(a,k*t))
```

Warning: I have unsuccessfully tried to prove this theorem. In the attached `.prf` file you will find a part of the proof, please take a brief look at it: The proof requires induction, I have achieved the base step and the induction step as hypotheses, therefore I think I managed to overcome most of the difficulties of the proving process. But I do not know how to instruct PVS to put the base step and the inductive step together and finish the proof. I browsed the PVS proof language manual, and tried to apply some concepts I found there, but with no results. Sorry.

I think that what I wrote is correct and that a more PVS-skilled person than me, like you, could easily finish it by starting at the point where a left it.

4.10 theorem x.

Informal semantics: this theorem states that under the same hypotheses as the previous theorem, in all future time instants the condition ϕ takes place within a time units.

$$\phi \wedge \mathrm{Alw}(\phi \ \Rightarrow \mathrm{Futr}(\phi, a)) \ \Rightarrow \mathrm{AlwF}(\mathrm{WithinF}(\phi, a))$$

A possible rewriting in TRIO PVS:

```
inverse_add_time_const2:LEMMA
  -t1 + t2 + t1 = t2

 theorem_x: THEOREM
  NOW( (a AND Alw(a IMPLIES Futr(a,t)))
     IMPLIES AlwF_i(WithinF_ii(a,t)))
```

Warning: this theorem, thanks to strong efforts and waste of time, has been almost proven. Thanks to the "inverse_add_time_const2" lemma which is here reported, I managed to prove the core of the semantics of the theorem. There's only a little type correctness condition left out, but I'm pretty sure that your experience will suggest a strategy to prove it or a way to slightly modify the starting hypotheses in order to completely prove the theorem. Please take a look at the proof details to see what I did and what I left out.

4.11 theorem xi.

Informal semantics: if an event occurs in a future instant a, it is true that from the time being up to that instant, there is a timeout, which is shorter than or equal to a, before which the event will occur.

$$\mathrm{Futr}(\phi, a)) \ \Rightarrow \mathrm{Lasts}(\mathrm{WithinF}((\phi, a), a)$$

Here is a possible PVS transposition of the theorem:

```
theorem_xi: THEOREM
  NOW(Futr(a,t) => Lasts_ii(WithinF_ii(a,t),t))
```

Warning:I have proven this theorem, and it required me some non-negligible efforts. Also this theorem is not simply provable via a grind, it instead requires a careful instantiation of the quantified formulas.

Please take a look at the proof log in the following section of this paper.

4.12 theorem xii.

Informal semantics: this theorem is useful in the anlisys of periodicity properties of nondeterministic systems. For instance, for some systems it can be the case that whenever an event ϕ occurs, after a time units, a further occurrence of ϕ will take place within b time units. In this case it can be proved that, if ϕ ever takes place, than from that time on, ϕ will always repeat within at most $a + b$ time units.

$$\phi \wedge \mathrm{Alw}(\phi \ \Rightarrow \mathrm{Futr}(\mathrm{WithinF}((\phi, b), a)) \ \Rightarrow \mathrm{AlwF}(\mathrm{WithinF}(\phi, a + b))$$

```
theorem_xii: THEOREM
  NOW( (a AND Alw(a IMPLIES Futr( WithinF_ii(a,p), q)))
      IMPLIES AlwF_e(WithinF_ii(a, p + q)))
```

Warning:like with theorem x, I nearly managed to prove this theorem. There's only a final TCC to prove I cannot get rid of. I'm sure you'll be able to dispose of if without effort.

4.13 theorem xiii.

Informal semantics: the theorem trivially follows from the properties of the temporal domain.

$\forall k$ Futr($\alpha \wedge$Lasted$(\alpha, a), a\ k)$ $\Rightarrow$ AlwF(α), $k \epsilon \mathbb{N}$

A possible PVS interpretation:

```
theorem_xiii: THEOREM
  NOW((FA! (n) : Futr(a AND Lasted_ii(a,t), t*n))
      IMPLIES NOW(AlwF_e(a)))
```

I could not prove this theorem. This is due to the fact that I could not find a suitable induction scheme. I tried both the weak "nat_induction" and the strong "NAT_induction" lemmas, with no success.

Please let me know how is it possible to prove this theorem, I am very interested in knowing how induction works in PVS in such a case.

4.14 theorem xiv.

Informal semantics: if an event occurs in the futur with a given timeout, it certainly satisfies any greater timeout.

WithinF(ϕ, t) $\Rightarrow$ $\forall s$ $(s > t \Rightarrow$WithinF$(\phi, s))$

A possible TRIO interpretation:

```
theorem_xiv: THEOREM
 NOW( WithinF_ee(a,t)
  IMPLIES (FA! (d): ( d > t IMPLIES WithinF_ee(a,d))))
```

A simple proof. Also *(grind)* is able to manage it.

4.15 theorem xv.

Informal semantics: the theorem simply states that if a timeout period is anticipated, then the timeout itself must be extended accordingly.
Futr(WithinF$(\phi, t_1), t_2)$ $\Rightarrow$ $\forall t$ $(0 < t < t_2 \Rightarrow$Futr(WithinF$(\phi, t_1 + t), t2 - t))$

In PVS:

```
inverse_add_time_const: LEMMA
  t1 + t2 - t1 = t2

theorem_xv: THEOREM
 NOW(Futr(WithinF_ee(a,t),d) AND t>0 IMPLIES
  FA! (s) : ((0<s AND s<d) IMPLIES Futr(WithinF_ee(a,t+s),d-s)))
```

Warning: theorem xv was a difficult theorem to prove for me, obviously (the *(grind)* rule is not effective on it). I had to introduce a dedicated lemma, called "inverse_add_time_const", not unlike theorem viii, prove it, and then use it to prove the theorem. Please take a look at the complete log of the proof, reported in the next section, since it contains an original contribution of mine.

5 Selected full-length TRIO PVS proofs details

Since this section is rather lengthy, I do not expect you to read it all. It should rather be used as a reference, to verify the details of what I stated in the previous section, and to see the points where failed to achieve the QED.

5.1 theorem i.

```
theorem_i :
  |-------
{1}    (FORALL (a: TD_Fmla, t: Time): Alw(LV(Alw(a)) IMPLIES Dist(a, t)))

Rule? (expand "Alw")
Expanding the definition of Alw, this simplifies to:
theorem_i :
  |-------
{1}    (FORALL (a: TD_Fmla, t: Time):
         FORALL (tt: Time):
           (LV(FORALL (tt: Time): a(tt)) IMPLIES Dist(a, t))(tt))

Rule? (expand "LV")
Expanding the definition of LV, this simplifies to:
theorem_i :
  |-------
{1}    (FORALL (a: TD_Fmla, t: Time):
         FORALL (tt: Time):
           ((LAMBDA (t: Time): FORALL (tt: Time): a(tt))
                IMPLIES Dist(a, t))(tt))

Rule? (expand "IMPLIES")
Expanding the definition of IMPLIES, this simplifies to:
theorem_i :
  |-------
{1}    (FORALL (a: TD_Fmla, t: Time):
         FORALL (tt: Time): (FORALL (tt: Time): a(tt)) IMPLIES Dist(a, t)(tt))

Rule? (expand "IMPLIES")
No change on: (EXPAND "IMPLIES")
theorem_i :
  |-------
{1}    (FORALL (a: TD_Fmla, t: Time):
         FORALL (tt: Time): (FORALL (tt: Time): a(tt)) IMPLIES Dist(a, t)(tt))
```

```
Rule? (skolem!)
Skolemizing, this simplifies to:
theorem_i :
  |-------
{1}    FORALL (tt: Time): (FORALL (tt: Time): a!1(tt)) IMPLIES Dist(a!1, t!1)(tt)

Rule? (ground)
Applying propositional simplification and decision procedures, this simplifies to:
theorem_i :
  |-------
[1]    FORALL (tt: Time): (FORALL (tt: Time): a!1(tt)) IMPLIES Dist(a!1, t!1)(tt)

Rule? (expand "Dist") this simplifies to:
theorem_i :
  |-------
{1}    FORALL (tt: Time): ((FORALL (tt: Time): a!1(tt)) IMPLIES a!1(tt + t!1))

Rule? (skolem!)
Expanding the definition of Dist,
Skolemizing, this simplifies to:
theorem_i :
  |-------
{1}    ((FORALL (tt: Time): a!1(tt)) IMPLIES a!1(tt!1 + t!1))

Rule? (expand "IMPLIES")
No change on: (EXPAND "IMPLIES")
theorem_i :
  |-------
{1}    ((FORALL (tt: Time): a!1(tt)) IMPLIES a!1(tt!1 + t!1))

Rule? (flatten)
Applying disjunctive simplification to flatten sequent, this simplifies to:
theorem_i :
{-1}    FORALL (tt: Time): a!1(tt)
  |-------
{1}    a!1(tt!1 + t!1)

Rule? (instantiate - "tt!1 + t!1")
Instantiating the top quantifier in - with the terms:  tt!1 + t!1, this simplifies to:
theorem_i :
{-1}    a!1(tt!1 + t!1)
  |-------
[1]    a!1(tt!1 + t!1)

which is trivially true.
Q.E.D.
```

5.2 theorem i.bis.

```
theorem_i_bis :
  |-------
{1}    (FORALL (a: TD_Fmla, t: Time): Alw(a) IMPLIES NOW(Dist(a, t)))

Rule? (expand "Alw")
Expanding the definition of Alw, this simplifies to:
theorem_i_bis :
  |-------
{1}    (FORALL (a: TD_Fmla, t: Time):
         ((FORALL (tt: Time): a(tt)) IMPLIES NOW(Dist(a, t))))

Rule? (expand "NOW")
Expanding the definition of NOW, this simplifies to:
theorem_i_bis :
  |-------
{1}    (FORALL (a: TD_Fmla, t: Time):
         ((FORALL (tt: Time): a(tt)) IMPLIES Dist(a, t)(0)))

Rule? (expand "Dist")
Expanding the definition of Dist, this simplifies to:
theorem_i_bis :
  |-------
{1}    (FORALL (a: TD_Fmla, t: Time): ((FORALL (tt: Time): a(tt)) IMPLIES a(t)))

Rule? (skolem!)
Skolemizing, this simplifies to:
theorem_i_bis :
  |-------
{1}    ((FORALL (tt: Time): a!1(tt)) IMPLIES a!1(t!1))

Rule? (skolem!)
No change on: (SKOLEM!)
theorem_i_bis :
  |-------
{1}    ((FORALL (tt: Time): a!1(tt)) IMPLIES a!1(t!1))

Rule? (flatten)
Applying disjunctive simplification to flatten sequent, this simplifies to:
theorem_i_bis :
{-1}    FORALL (tt: Time): a!1(tt)
  |-------
{1}    a!1(t!1)

Rule? (instantiate -1 "t!1")
Instantiating the top quantifier in -1 with the terms:  t!1, this simplifies to:
theorem_i_bis :
{-1}    a!1(t!1)
  |-------
```

```
[1]    a!1(t!1)
```

which is trivially true.
Q.E.D.

5.3 theorem i.ter.

```
theorem_i_ter :
  |-------
{1}    (FORALL (a: TD_Fmla, t: Time): NOW(LV(Alw(a)) IMPLIES Dist(a, t)))

Rule? (skolem!)
Skolemizing, this simplifies to:
theorem_i_ter :
  |-------
{1}    NOW(LV(Alw(a!1)) IMPLIES Dist(a!1, t!1))

Rule? (expand "NOW")
Expanding the definition of NOW, this simplifies to:
theorem_i_ter :
  |-------
{1}    (LV(Alw(a!1)) IMPLIES Dist(a!1, t!1))(0)

Rule? (expand "LV")
Expanding the definition of LV, this simplifies to:
theorem_i_ter :
  |-------
{1}    ((LAMBDA (t: Time): Alw(a!1)) IMPLIES Dist(a!1, t!1))(0)

Rule? (expand "Alw")
Expanding the definition of Alw, this simplifies to:
theorem_i_ter :
  |-------
{1}    ((LAMBDA (t: Time): FORALL (tt: Time): a!1(tt)) IMPLIES Dist(a!1, t!1))(0)

Rule? (expand "IMPLIES")
Expanding the definition of IMPLIES, this simplifies to:
theorem_i_ter :
  |-------
{1}    (FORALL (tt: Time): a!1(tt)) IMPLIES Dist(a!1, t!1)(0)

Rule? (flatten)
Applying disjunctive simplification to flatten sequent, this simplifies to:
theorem_i_ter :
{-1}    FORALL (tt: Time): a!1(tt)
  |-------
{1}    Dist(a!1, t!1)(0)

Rule? (expand "Dist")
Expanding the definition of Dist, this simplifies to:
```

```
theorem_i_ter :
[-1]    FORALL (tt: Time): a!1(tt)
  |-------
{1}    a!1(t!1)

Rule? (instantiate -1 "t!1")
Instantiating the top quantifier in -1 with the terms:  t!1, this simplifies to:
theorem_i_ter :
{-1}    a!1(t!1)
  |-------
[1]    a!1(t!1)

which is trivially true.
Q.E.D.
```

5.4 theorem ii.

```
theorem_ii :
  |-------
{1}    (FORALL (a: TD_Fmla): Alw(LV(Alw(a)) IMPLIES a))

Rule? (skolem!)
Skolemizing, this simplifies to:
theorem_ii :
  |-------
{1}    Alw(LV(Alw(a!1)) IMPLIES a!1)

Rule? (expand "Alw")
Expanding the definition of Alw, this simplifies to:
theorem_ii :
  |-------
{1}    FORALL (tt: Time): (LV(FORALL (tt: Time): a!1(tt)) IMPLIES a!1)(tt)

Rule? (skolem!)
Skolemizing, this simplifies to:
theorem_ii :
  |-------
{1}    (LV(FORALL (tt: Time): a!1(tt)) IMPLIES a!1)(tt!1)

Rule? (expand "LV")
Expanding the definition of LV, this simplifies to:
theorem_ii :
  |-------
{1}    ((LAMBDA (t: Time): FORALL (tt: Time): a!1(tt)) IMPLIES a!1)(tt!1)

Rule? (expand "IMPLIES")
Expanding the definition of IMPLIES, this simplifies to:
theorem_ii :
  |-------
{1}    (FORALL (tt: Time): a!1(tt)) IMPLIES a!1(tt!1)
```

```
Rule? (flatten)
Applying disjunctive simplification to flatten sequent, this simplifies to:
theorem_ii :
{-1}    FORALL (tt: Time): a!1(tt)
  |-------
{1}    a!1(tt!1)

Rule? (instantiate -1 "tt!1")
Instantiating the top quantifier in -1 with the terms:  tt!1, this simplifies to:
theorem_ii :
{-1}    a!1(tt!1)
  |-------
[1]    a!1(tt!1)

which is trivially true.
Q.E.D.
```

5.5 theorem ii.bis.

A very simple *(grind)*-based proof.

```
theorem_ii_bis :
  |-------
{1}    (FORALL (a: TD_Fmla): Alw(a) IMPLIES NOW(a))

Rule? (grind)
Alw rewrites Alw(a)
  to  FORALL (tt: Time): a(tt)
NOW rewrites NOW(a)
  to  a(0)
Trying repeated skolemization, instantiation, and if-lifting,
Q.E.D.
```

5.6 theorem iii.

```
theorem_iii :
  |-------
{1}    (FORALL (a: TD_Fmla, b: TD_Fmla, t: Time):
         Alw((LV(Alw(a IMPLIES b)) AND Dist(a, t)) IMPLIES Dist(b, t)))

Rerunning step: (EXPAND "Alw")
Expanding the definition of Alw, this simplifies to:
theorem_iii :
  |-------
{1}    (FORALL (a: TD_Fmla, b: TD_Fmla, t: Time):
         FORALL (tt: Time):
           ((LV(FORALL (tt: Time): (a IMPLIES b)(tt)) AND Dist(a, t))
                IMPLIES Dist(b, t))(tt))
```

```
Rerunning step: (EXPAND "LV")
Expanding the definition of LV, this simplifies to:
theorem_iii :
  |-------
{1}    (FORALL (a: TD_Fmla, b: TD_Fmla, t: Time):
         FORALL (tt: Time):
           (((LAMBDA (t: Time): FORALL (tt: Time): (a IMPLIES b)(tt))
                 AND Dist(a, t))
                IMPLIES Dist(b, t))(tt))

Rerunning step: (EXPAND "IMPLIES")
Expanding the definition of IMPLIES, this simplifies to:
theorem_iii :

  |-------
{1}    (FORALL (a: TD_Fmla, b: TD_Fmla, t: Time):
         FORALL (tt: Time):
           ((LAMBDA (t: Time): FORALL (tt: Time): a(tt) IMPLIES b(tt))
                AND Dist(a, t))(tt)
               IMPLIES Dist(b, t)(tt))

Rerunning step: (SKOLEM!)
Skolemizing, this simplifies to:
theorem_iii :
  |-------
{1}    FORALL (tt: Time):
        ((LAMBDA (t: Time): FORALL (tt: Time): a!1(tt) IMPLIES b!1(tt))
             AND Dist(a!1, t!1))(tt)
            IMPLIES Dist(b!1, t!1)(tt)

Rerunning step: (SKOLEM!)
Skolemizing, this simplifies to:
theorem_iii :
  |-------
{1}    ((LAMBDA (t: Time): FORALL (tt: Time): a!1(tt) IMPLIES b!1(tt))
           AND Dist(a!1, t!1))(tt!1)
          IMPLIES Dist(b!1, t!1)(tt!1)

Rerunning step: (EXPAND "Dist")
Expanding the definition of Dist, this simplifies to:
theorem_iii :
  |-------
{1}    (((LAMBDA (t: Time): FORALL (tt: Time): a!1(tt) IMPLIES b!1(tt))
            AND (LAMBDA (t: Time): a!1(t + t!1)))(tt!1)
           IMPLIES b!1(t!1 + tt!1))

Rerunning step: (FLATTEN)
Applying disjunctive simplification to flatten sequent, this simplifies to:
theorem_iii :
{-1}    ((LAMBDA (t: Time): FORALL (tt: Time): a!1(tt) IMPLIES b!1(tt))
```

```
          AND (LAMBDA (t: Time): a!1(t + t!1)))(tt!1)
  |-------
{1}    b!1(t!1 + tt!1)

Rerunning step: (GRIND)
AND rewrites
   ((LAMBDA (t: Time): FORALL (tt: Time): a!1(tt) IMPLIES b!1(tt))
       AND (LAMBDA (t: Time): a!1(t + t!1)))(tt!1)
  to  (FORALL (tt: Time): a!1(tt) IMPLIES b!1(tt)) AND a!1(t!1 + tt!1)
Trying repeated skolemization, instantiation, and if-lifting,
Q.E.D.
```

5.7 theorem iii.bis.

```
theorem_iii_bis :
  |-------
{1}    (FORALL (a: TD_Fmla, b: TD_Fmla, t: Time):
         Alw(a IMPLIES b) AND NOW(Dist(a, t)) IMPLIES NOW(Dist(b, t)))

Rule? (expand "Alw")
Expanding the definition of Alw, this simplifies to:
theorem_iii_bis :
  |-------
{1}    (FORALL (a: TD_Fmla, b: TD_Fmla, t: Time):
         (((FORALL (tt: Time): (a IMPLIES b)(tt)) AND NOW(Dist(a, t)))
              IMPLIES NOW(Dist(b, t))))

Rule? (expand "NOW")
Expanding the definition of NOW, this simplifies to:
theorem_iii_bis :
  |-------
{1}    (FORALL (a: TD_Fmla, b: TD_Fmla, t: Time):
         (((FORALL (tt: Time): (a IMPLIES b)(tt)) AND Dist(a, t)(0))
              IMPLIES Dist(b, t)(0)))

Rule? (expand "Dist")
Expanding the definition of Dist, this simplifies to:
theorem_iii_bis :
  |-------
{1}    (FORALL (a: TD_Fmla, b: TD_Fmla, t: Time):
         (((FORALL (tt: Time): (a IMPLIES b)(tt)) AND a(t)) IMPLIES b(t)))

Rule? (skolem!)
Skolemizing, this simplifies to:
theorem_iii_bis :
  |-------
{1}    (((FORALL (tt: Time): (a!1 IMPLIES b!1)(tt)) AND a!1(t!1)) IMPLIES b!1(t!1))

Rule? (flatten)
Applying disjunctive simplification to flatten sequent, this simplifies to:
```

```
theorem_iii_bis :
{-1}    FORALL (tt: Time): (a!1 IMPLIES b!1)(tt)
{-2}    a!1(t!1)
  |-------
{1}    b!1(t!1)

Rule? (instantiate -1 "t!1")
Instantiating the top quantifier in -1 with the terms:  t!1, this simplifies to:
theorem_iii_bis :
{-1}    (a!1 IMPLIES b!1)(t!1)
[-2]    a!1(t!1)
  |-------
[1]    b!1(t!1)

Rule? (expand "IMPLIES")
Expanding the definition of IMPLIES, this simplifies to:
theorem_iii_bis :
{-1}    FALSE
[-2]    a!1(t!1)
  |-------
[1]    b!1(t!1)

which is trivially true.
Q.E.D.
```

5.8 theorem iv.

```
theorem_iv :
  |-------
{1}    (FORALL (a: TD_Fmla, t: Time):
         Alw(((a IMPLIES LV(Alw(a)))
                  AND ((NOT a) IMPLIES (LV((Alw(NOT a)))))
                    AND (Dist(a, t)))
                 IMPLIES a))

Rule? (trio-simp)
elim_IMPLIES rewrites (a IMPLIES LV(Alw(a)))
  to  (LV(Alw(a)) OR NOT a)
elim_IMPLIES rewrites ((NOT a) IMPLIES (LV((Alw(NOT a)))))
  to  ((LV((Alw(NOT a)))) OR NOT (NOT a))
bury_NOT_2 rewrites NOT (LV(Alw(a)) OR NOT a)
  to  (NOT LV(Alw(a)) OR NOT NOT a)
bury_NOT_2 rewrites NOT ((LV((Alw(NOT a)))) OR NOT (NOT a))
  to  (NOT (LV((Alw(NOT a)))) OR NOT NOT (NOT a))
bury_NOT_1 rewrites
   NOT (((LV((Alw(NOT a)))) OR NOT (NOT a)) AND (Dist(a, t)))
  to  ((NOT (LV((Alw(NOT a)))) OR NOT NOT (NOT a)) OR NOT (Dist(a, t)))
left_group_OR rewrites
   ((NOT LV(Alw(a)) OR NOT NOT a) OR (NOT (LV((Alw(NOT a)))) OR NOT NOT (NOT a)))
  to  (((NOT LV(Alw(a)) OR NOT NOT a) OR NOT (LV((Alw(NOT a)))))
```

```
          OR NOT NOT (NOT a))
left_group_OR rewrites
   ((NOT LV(Alw(a)) OR NOT NOT a)
       OR ((NOT (LV((Alw(NOT a)))) OR NOT NOT (NOT a)) OR NOT (Dist(a, t))))
  to  ((((NOT LV(Alw(a)) OR NOT NOT a) OR NOT (LV((Alw(NOT a)))))
           OR NOT NOT (NOT a))
          OR NOT (Dist(a, t)))
bury_NOT_1 rewrites
   NOT
    ((LV(Alw(a)) OR NOT a)
         AND ((LV((Alw(NOT a)))) OR NOT (NOT a)) AND (Dist(a, t)))
  to  ((((NOT LV(Alw(a)) OR NOT NOT a) OR NOT (LV((Alw(NOT a)))))
           OR NOT NOT (NOT a))
          OR NOT (Dist(a, t)))
left_group_OR rewrites (a OR (NOT LV(Alw(a)) OR NOT NOT a))
  to  ((a OR NOT LV(Alw(a))) OR NOT NOT a)
left_group_OR rewrites
   (a OR ((NOT LV(Alw(a)) OR NOT NOT a) OR NOT (LV((Alw(NOT a))))))
  to  (((a OR NOT LV(Alw(a))) OR NOT NOT a) OR NOT (LV((Alw(NOT a)))))
left_group_OR rewrites
   (a
       OR
     (((NOT LV(Alw(a)) OR NOT NOT a) OR NOT (LV((Alw(NOT a)))))
         OR NOT NOT (NOT a)))
  to  ((((a OR NOT LV(Alw(a))) OR NOT NOT a) OR NOT (LV((Alw(NOT a)))))
          OR NOT NOT (NOT a))
left_group_OR rewrites
   (a
       OR
     ((((NOT LV(Alw(a)) OR NOT NOT a) OR NOT (LV((Alw(NOT a)))))
          OR NOT NOT (NOT a))
         OR NOT (Dist(a, t))))
  to  (((((a OR NOT LV(Alw(a))) OR NOT NOT a) OR NOT (LV((Alw(NOT a)))))
           OR NOT NOT (NOT a))
          OR NOT (Dist(a, t)))
elim_IMPLIES rewrites
   ((LV(Alw(a)) OR NOT a)
       AND ((LV((Alw(NOT a)))) OR NOT (NOT a)) AND (Dist(a, t)))
      IMPLIES a
  to  (((((a OR NOT LV(Alw(a))) OR NOT NOT a) OR NOT (LV((Alw(NOT a)))))
           OR NOT NOT (NOT a))
          OR NOT (Dist(a, t)))
Simplifying temporal quantifications, this simplifies to:
theorem_iv :
  |-------
{1}    (FORALL (a: TD_Fmla, t: Time):
         Alw((((((a OR NOT LV(Alw(a))) OR NOT NOT a) OR NOT (LV((Alw(NOT a)))))
                  OR NOT NOT (NOT a))
                 OR NOT (Dist(a, t)))))
```

```
Rule? (skosimp)
Skolemizing and flattening, this simplifies to:
theorem_iv :
  |-------
{1}    Alw((((((a!1 OR NOT LV(Alw(a!1))) OR NOT NOT a!1)
                 OR NOT (LV((Alw(NOT a!1)))))
                OR NOT NOT (NOT a!1))
               OR NOT (Dist(a!1, t!1))))

Rule? (grind)
Alw rewrites Alw(a!1)
  to  FORALL (tt: Time): a!1(tt)
NOT rewrites (NOT a!1)(tt)
  to  NOT a!1(tt)
Alw rewrites (Alw(NOT a!1))
  to  FORALL (tt: Time): NOT a!1(tt)
LV rewrites LV(FORALL (tt: Time): a!1(tt))(tt)
  to  FORALL (tt: Time): a!1(tt)
NOT rewrites (NOT LV(FORALL (tt: Time): a!1(tt)))(tt)
  to  NOT (FORALL (tt: Time): a!1(tt))
OR rewrites (a!1 OR NOT LV(FORALL (tt: Time): a!1(tt)))(tt)
  to  a!1(tt) OR NOT (FORALL (tt: Time): a!1(tt))
NOT rewrites (NOT a!1)(tt)
  to  NOT a!1(tt)
NOT rewrites (NOT NOT a!1)(tt)
  to  a!1(tt)
OR rewrites ((a!1 OR NOT LV(FORALL (tt: Time): a!1(tt))) OR NOT NOT a!1)(tt)
  to  (a!1(tt) OR NOT (FORALL (tt: Time): a!1(tt))) OR a!1(tt)
LV rewrites (LV(FORALL (tt: Time): NOT a!1(tt)))(tt)
  to  FORALL (tt: Time): NOT a!1(tt)
NOT rewrites (NOT (LV(FORALL (tt: Time): NOT a!1(tt))))(tt)
  to  NOT (FORALL (tt: Time): NOT a!1(tt))
OR rewrites
   (((a!1 OR NOT LV(FORALL (tt: Time): a!1(tt))) OR NOT NOT a!1)
       OR NOT (LV(FORALL (tt: Time): NOT a!1(tt))))(tt)
  to  ((a!1(tt) OR NOT (FORALL (tt: Time): a!1(tt))) OR a!1(tt))
          OR NOT (FORALL (tt: Time): NOT a!1(tt))
NOT rewrites (NOT NOT (NOT a!1))(tt)
  to  NOT a!1(tt)
OR rewrites
   ((((a!1 OR NOT LV(FORALL (tt: Time): a!1(tt))) OR NOT NOT a!1)
        OR NOT (LV(FORALL (tt: Time): NOT a!1(tt))))
       OR NOT NOT (NOT a!1))(tt)
  to  (((a!1(tt) OR NOT (FORALL (tt: Time): a!1(tt))) OR a!1(tt))
           OR NOT (FORALL (tt: Time): NOT a!1(tt)))
          OR NOT a!1(tt)
Dist rewrites (Dist(a!1, t!1))(tt)
  to  a!1(tt + t!1)
NOT rewrites (NOT (Dist(a!1, t!1)))(tt)
  to  NOT a!1(tt + t!1)
```

```
OR rewrites
   ((((a!1 OR NOT LV(FORALL (tt: Time): a!1(tt))) OR NOT NOT a!1)
         OR NOT (LV(FORALL (tt: Time): NOT a!1(tt))))
        OR NOT NOT (NOT a!1))
       OR NOT (Dist(a!1, t!1)))(tt)
  to  ((((a!1(tt) OR NOT (FORALL (tt: Time): a!1(tt))) OR a!1(tt))
           OR NOT (FORALL (tt: Time): NOT a!1(tt)))
          OR NOT a!1(tt))
         OR NOT a!1(tt + t!1)
Alw rewrites
   Alw(((((a!1 OR NOT LV(FORALL (tt: Time): a!1(tt))) OR NOT NOT a!1)
             OR NOT (LV(FORALL (tt: Time): NOT a!1(tt))))
            OR NOT NOT (NOT a!1))
           OR NOT (Dist(a!1, t!1))))
  to  FORALL (tt: Time):
        ((((a!1(tt) OR NOT (FORALL (tt: Time): a!1(tt))) OR a!1(tt))
             OR NOT (FORALL (tt: Time): NOT a!1(tt)))
            OR NOT a!1(tt))
           OR NOT a!1(tt + t!1)
Trying repeated skolemization, instantiation, and if-lifting,
Q.E.D.
```

5.9 theorem v.bis.

A very simple *(grind)*-based proof.

```
theorem_v_bis :
  |-------
{1}    (FORALL (a: TD_Fmla, b: TD_Fmla, t: Time):
         NOW(Futr(a IMPLIES b, t) IMPLIES (Futr(a, t) IMPLIES Futr(b, t))))

Rule? (grind)
IMPLIES rewrites (a IMPLIES b)(t)
  to  a(t) IMPLIES b(t)
Futr rewrites Futr(a IMPLIES b, t)(0)
  to  a(t) IMPLIES b(t)
Futr rewrites Futr(a, t)(0)
  to  a(t)
Futr rewrites Futr(b, t)(0)
  to  b(t)
IMPLIES rewrites (Futr(a, t) IMPLIES Futr(b, t))(0)
  to  a(t) IMPLIES b(t)
IMPLIES rewrites
   (Futr(a IMPLIES b, t) IMPLIES (Futr(a, t) IMPLIES Futr(b, t)))(0)
  to  (a(t) IMPLIES b(t)) IMPLIES a(t) IMPLIES b(t)
NOW rewrites
   NOW(Futr(a IMPLIES b, t) IMPLIES (Futr(a, t) IMPLIES Futr(b, t)))
  to  (a(t) IMPLIES b(t)) IMPLIES a(t) IMPLIES b(t)
Trying repeated skolemization, instantiation, and if-lifting,
Q.E.D.
```

5.10 theorem vi.b.original

This is the original version of the theorem, as reported on page 11 of the original paper; as told before, in this form the theorem is false.

```
theorem_vi_b_original :
  |-------
{1}    (FORALL (a: TD_Fmla, t1: Time, t2: Time):
         NOW(Futr(Past(a, t1), t2) AND LV(t1 > t2) IMPLIES Futr(a, t1 - t2)))

Rule? (grind)
Past rewrites Past(a, t1)(t2)
  to  a(t2 - t1)
Futr rewrites Futr(Past(a, t1), t2)(0)
  to  a(t2 - t1)
LV rewrites LV(t1 > t2)(0)
  to  t1 > t2
AND rewrites (Futr(Past(a, t1), t2) AND LV(t1 > t2))(0)
  to  a(t2 - t1) AND t1 > t2
Futr rewrites Futr(a, t1 - t2)(0)
  to  a(t1 - t2)
IMPLIES rewrites
   (Futr(Past(a, t1), t2) AND LV(t1 > t2) IMPLIES Futr(a, t1 - t2))(0)
  to  a(t2 - t1) AND t1 > t2 IMPLIES a(t1 - t2)
NOW rewrites
   NOW(Futr(Past(a, t1), t2) AND LV(t1 > t2) IMPLIES Futr(a, t1 - t2))
  to  a(t2 - t1) AND t1 > t2 IMPLIES a(t1 - t2)
Trying repeated skolemization, instantiation, and if-lifting,
this simplifies to:
theorem_vi_b_original :

{-1}    real_pred(t1!1)
{-2}    real_pred(t2!1)
{-3}    a!1(t2!1 - t1!1)
{-4}    t1!1 > t2!1
  |-------
{1}    a!1(t1!1 - t2!1)

Rule? (fail)

Attempted proof of theorem_vi_b_original failed.
Attempted proof of theorem_vi_b_original failed.
Failed!
```

Nothing can be done save to fail, since it is clear that with antecedent (-3) it will never be possible to prove the consequent.

5.11 theorem vi.b.correct.

```
theorem_vi_b_correct :
  |-------
```

```
{1}    (FORALL (a: TD_Fmla, t1: Time, t2: Time):
         NOW(Futr(Past(a, t1), t2) AND LV(t1 > t2) IMPLIES Futr(a, t2 - t1)))

Rule? (expand "Futr")
Expanding the definition of Futr, this simplifies to:
theorem_vi_b_correct :
  |-------
{1}    (FORALL (a: TD_Fmla, t1: Time, t2: Time):
         NOW((((LAMBDA (t: Time): Past(a, t1)(t2 + t)) AND LV(t1 > t2))
                  IMPLIES (LAMBDA (t: Time): a(t2 + t - t1)))))

Rule? (expand "Past")
Expanding the definition of Past, this simplifies to:
theorem_vi_b_correct :
  |-------
{1}    (FORALL (a: TD_Fmla, t1: Time, t2: Time):
         NOW((((LAMBDA (t: Time): a(t2 + t - t1)) AND LV(t1 > t2))
                  IMPLIES (LAMBDA (t: Time): a(t2 + t - t1)))))

Rule? (expand "LV")
Expanding the definition of LV, this simplifies to:
theorem_vi_b_correct :
  |-------
{1}    (FORALL (a: TD_Fmla, t1: Time, t2: Time):
         NOW((((LAMBDA (t: Time): a(t2 + t - t1))
                   AND (LAMBDA (t: Time): t1 > t2))
                  IMPLIES (LAMBDA (t: Time): a(t2 + t - t1)))))

Rule? (skolem!)
Skolemizing, this simplifies to:
theorem_vi_b_correct :
  |-------
{1}    NOW((((LAMBDA (t: Time): a!1(t2!1 + t - t1!1))
                AND (LAMBDA (t: Time): t1!1 > t2!1))
               IMPLIES (LAMBDA (t: Time): a!1(t2!1 + t - t1!1))))

Rule? (expand "NOW")
Expanding the definition of NOW, this simplifies to:
theorem_vi_b_correct :
  |-------
{1}    (((LAMBDA (t: Time): a!1(t - t1!1 + t2!1))
            AND (LAMBDA (t: Time): t1!1 > t2!1))
           IMPLIES (LAMBDA (t: Time): a!1(t - t1!1 + t2!1)))(0)

Rule? (expand "AND")
Expanding the definition of AND, this simplifies to:
theorem_vi_b_correct :
  |-------
{1}    ((LAMBDA (t: Time): a!1(t - t1!1 + t2!1) AND t1!1 > t2!1)
           IMPLIES (LAMBDA (t: Time): a!1(t - t1!1 + t2!1)))(0)
```

```
Rule? (expand "IMPLIES")
Expanding the definition of IMPLIES, this simplifies to:
theorem_vi_b_correct :
  |-------
{1}    a!1(t2!1 - t1!1) AND t1!1 > t2!1 IMPLIES a!1(t2!1 - t1!1)

Rule? (flatten)
Applying disjunctive simplification to flatten sequent, this simplifies to:
theorem_vi_b_correct :

{-1}    a!1(t2!1 - t1!1)
{-2}    t1!1 > t2!1
  |-------
{1}    a!1(t2!1 - t1!1)

which is trivially true.
Q.E.D.
```

Et voilà!

5.12 theorem vii.c.original

This is the original version of the theorem, as reported on page 11 of the original paper; as told before, not unlike the previous case, in this form the theorem is false.

```
theorem_vi_c_original :
  |-------
{1}    (FORALL (a: TD_Fmla, t1: Time, t2: Time):
         NOW(Futr(Past(a, t1), t2) AND LV(t1 < t2) IMPLIES Past(a, t2 - t1)))

Rule? (grind)
Past rewrites Past(a, t1)(t2)
  to  a(t2 - t1)
Futr rewrites Futr(Past(a, t1), t2)(0)
  to  a(t2 - t1)
LV rewrites LV(t1 < t2)(0)
  to  t1 < t2
AND rewrites (Futr(Past(a, t1), t2) AND LV(t1 < t2))(0)
  to  a(t2 - t1) AND t1 < t2
Past rewrites Past(a, t2 - t1)(0)
  to  a(t1 - t2)
IMPLIES rewrites
   (Futr(Past(a, t1), t2) AND LV(t1 < t2) IMPLIES Past(a, t2 - t1))(0)
  to  a(t2 - t1) AND t1 < t2 IMPLIES a(t1 - t2)
NOW rewrites
   NOW(Futr(Past(a, t1), t2) AND LV(t1 < t2) IMPLIES Past(a, t2 - t1))
  to  a(t2 - t1) AND t1 < t2 IMPLIES a(t1 - t2)
Trying repeated skolemization, instantiation, and if-lifting,
```

```
this simplifies to:
theorem_vi_c_original :
{-1}    real_pred(t1!1)
{-2}    real_pred(t2!1)
{-3}    a!1(t2!1 - t1!1)
{-4}    t1!1 < t2!1
  |-------
{1}    a!1(t1!1 - t2!1)

Rule? (fail)
Attempted proof of theorem_vi_c_original failed.
Attempted proof of theorem_vi_c_original failed.
Failed!

Run time  = 1.03 secs.
Real time = 10.58 secs.
NIL
```

Again we have to fail, since the (-3) antecedent clearly conflicts with consequent (1).

5.13 theorem vii.c.correct

```
theorem_vi_c_correct :
  |-------
{1}    (FORALL (a: TD_Fmla, t1: Time, t2: Time):
         NOW(Futr(Past(a, t1), t2) AND LV(t1 < t2) IMPLIES Past(a, t1 - t2)))

Rule? (expand "Futr")
Expanding the definition of Futr, this simplifies to:
theorem_vi_c_correct :
  |-------
{1}    (FORALL (a: TD_Fmla, t1: Time, t2: Time):
         NOW((((LAMBDA (t: Time): Past(a, t1)(t2 + t)) AND LV(t1 < t2))
                  IMPLIES Past(a, t1 - t2))))

Rule? (expand "NOW")
Expanding the definition of NOW, this simplifies to:
theorem_vi_c_correct :
  |-------
{1}    (FORALL (a: TD_Fmla, t1: Time, t2: Time):
         (((LAMBDA (t: Time): Past(a, t1)(t2 + t)) AND LV(t1 < t2))
              IMPLIES Past(a, t1 - t2))(0))

Rule? (expand "Past")
Expanding the definition of Past, this simplifies to:
theorem_vi_c_correct :
  |-------
{1}    (FORALL (a: TD_Fmla, t1: Time, t2: Time):
         (((LAMBDA (t: Time): a(t2 + t - t1)) AND LV(t1 < t2))
```

```
              IMPLIES (LAMBDA (t: Time): a(t2 + t - t1)))(0))

Rule? (skolem!)
Skolemizing, this simplifies to:
theorem_vi_c_correct :
  |-------
{1}    (((LAMBDA (t: Time): a!1(t2!1 + t - t1!1)) AND LV(t1!1 < t2!1))
           IMPLIES (LAMBDA (t: Time): a!1(t2!1 + t - t1!1)))(0)

Rule? (expand "LV")
Expanding the definition of LV, this simplifies to:
theorem_vi_c_correct :
  |-------
{1}    (((LAMBDA (t: Time): a!1(t - t1!1 + t2!1))
            AND (LAMBDA (t: Time): t1!1 < t2!1))
           IMPLIES (LAMBDA (t: Time): a!1(t - t1!1 + t2!1)))(0)

Rule? (expand "AND")
Expanding the definition of AND, this simplifies to:
theorem_vi_c_correct :
  |-------
{1}    ((LAMBDA (t: Time): a!1(t - t1!1 + t2!1) AND t1!1 < t2!1)
           IMPLIES (LAMBDA (t: Time): a!1(t - t1!1 + t2!1)))(0)

Rule? (expand "IMPLIES")
Expanding the definition of IMPLIES, this simplifies to:
theorem_vi_c_correct :
  |-------
{1}    a!1(t2!1 - t1!1) AND t1!1 < t2!1 IMPLIES a!1(t2!1 - t1!1)

Rule? (flatten)
Applying disjunctive simplification to flatten sequent, this simplifies to:
theorem_vi_c_correct :
{-1}    a!1(t2!1 - t1!1)
{-2}    t1!1 < t2!1
  |-------
{1}    a!1(t2!1 - t1!1)

which is trivially true.
Q.E.D.
```

Et voilà encore !

5.14 theorem vii.a.

```
theorem_vii_a :
  |-------
{1}    (FORALL (a: TD_Fmla, t: Time):
         NOW(Dist(FA! (x: int): a, t) IMPLIES FA! (x: int): Dist(a, t)))
```

```
Rule? (expand "Dist")
Expanding the definition of Dist, this simplifies to:
theorem_vii_a :
  |-------
{1}    (FORALL (a: TD_Fmla, t: Time):
         NOW(((LAMBDA (t_60: Time): FA! (x: int): a(t + t_60))
                  IMPLIES FA! (x: int): LAMBDA (t_61: Time): a(t + t_61))))

Rule? (skolem!)
Skolemizing, this simplifies to:
theorem_vii_a :
  |-------
{1}    NOW(((LAMBDA (t_60: Time): FA! (x: int): a!1(t!1 + t_60))
               IMPLIES FA! (x: int): LAMBDA (t_61: Time): a!1(t!1 + t_61)))

Rule? (expand "NOW")
Expanding the definition of NOW, this simplifies to:
theorem_vii_a :
  |-------
{1}    ((LAMBDA (t_60: Time): FA! (x: int): a!1(t_60 + t!1))
           IMPLIES FA! (x: int): LAMBDA (t_61: Time): a!1(t_61 + t!1))(0)

Rule? (expand "IMPLIES")
Expanding the definition of IMPLIES, this simplifies to:
theorem_vii_a :
  |-------
{1}    FA! (x: int): a!1(t!1)
          IMPLIES FA! (x: int): LAMBDA (t_61: Time): a!1(t_61 + t!1)(0)

Rule? (expand "FA")
Expanding the definition of FA, this simplifies to:
theorem_vii_a :
  |-------
{1}    ((FORALL (x: int): a!1(t!1)) IMPLIES (FORALL (x: int): a!1(t!1)))

Rule? (flatten)
Applying disjunctive simplification to flatten sequent, this simplifies to:
theorem_vii_a :
{-1}    FORALL (x: int): a!1(t!1)
  |-------
{1}    FORALL (x: int): a!1(t!1)

which is trivially true.
Q.E.D.
```

5.15 theorem vii.a.rev.

```
theorem_vii_a_rev :
  |-------
{1}    (FORALL (a: TD_Fmla, t: Time):
```

```
         NOW(FA! (x: int): Dist(a, t) IMPLIES Dist(FA! (x: int): a, t)))

Rule? (skolem!)
Skolemizing, this simplifies to:
theorem_vii_a_rev :

  |-------
{1}    NOW(FA! (x: int): Dist(a!1, t!1) IMPLIES Dist(FA! (x: int): a!1, t!1))

Rule? (expand "NOW")
Expanding the definition of NOW, this simplifies to:
theorem_vii_a_rev :
  |-------
{1}    FA! (x: int): Dist(a!1, t!1) IMPLIES Dist(FA! (x: int): a!1, t!1)(0)

Rule? (expand "Dist")
Expanding the definition of Dist, this simplifies to:
theorem_vii_a_rev :
  |-------
{1}    FA! (x: int):
        ((LAMBDA (t: Time): a!1(t + t!1))
             IMPLIES (LAMBDA (t: Time): FA! (x: int): a!1(t + t!1)))(0)

Rule? (expand "FA")
Expanding the definition of FA, this simplifies to:
theorem_vii_a_rev :
  |-------
{1}    FORALL (x: int):
        ((LAMBDA (t: Time): a!1(t + t!1))
             IMPLIES (LAMBDA (t: Time): FORALL (x: int): a!1(t + t!1)))(0)

Rule? (skolem!)
Skolemizing, this simplifies to:
theorem_vii_a_rev :
  |-------
{1}    ((LAMBDA (t: Time): a!1(t + t!1))
           IMPLIES (LAMBDA (t: Time): FORALL (x: int): a!1(t + t!1)))(0)

Rule? (expand "IMPLIES")
Expanding the definition of IMPLIES, this simplifies to:
theorem_vii_a_rev :
  |-------
{1}    a!1(t!1) IMPLIES (FORALL (x: int): a!1(t!1))

Rule? (flatten)
Applying disjunctive simplification to flatten sequent,
this simplifies to:
theorem_vii_a_rev :
{-1}    a!1(t!1)
  |-------
```

```
{1}    FORALL (x: int): a!1(t!1)

Rule? (skolem!)
Skolemizing,
Q.E.D.
```

5.16 theorem vii.b.rev

```
theorem_vii_b_rev :
  |-------
{1}    (FORALL (a: TD_Fmla, t: Time):
         NOW(EX! (x: int): Dist(a, t)) IMPLIES NOW(Dist(EX! (x: int): a, t)))

Rule? (skolem!)
Skolemizing, this simplifies to:
theorem_vii_b_rev :
  |-------
{1}    NOW(EX! (x: int): Dist(a!1, t!1))
          IMPLIES NOW(Dist(EX! (x: int): a!1, t!1))

Rule? (flatten)
Applying disjunctive simplification to flatten sequent, this simplifies to:
theorem_vii_b_rev :
{-1}    NOW(EX! (x: int): Dist(a!1, t!1))
  |-------
{1}    NOW(Dist(EX! (x: int): a!1, t!1))

Rule? (expand "Dist")
Expanding the definition of Dist, this simplifies to:
theorem_vii_b_rev :
{-1}    NOW(EX! (x: int): LAMBDA (t: Time): a!1(t + t!1))
  |-------
{1}    NOW(LAMBDA (t: Time): EX! (x: int): a!1(t + t!1))

Rule? (expand "NOW")
Expanding the definition of NOW, this simplifies to:
theorem_vii_b_rev :
{-1}    EX! (x: int): LAMBDA (t: Time): a!1(t + t!1)(0)
  |-------
{1}    EX! (x: int): a!1(t!1)

Rule? (expand "EX")
Expanding the definition of EX, this simplifies to:
theorem_vii_b_rev :
{-1}    EXISTS (x: int): a!1(t!1)
  |-------
{1}    EXISTS (x: int): a!1(t!1)

which is trivially true.
Q.E.D.
```

5.17 theorem viii.

```
theorem_viii :
  |-------
{1}    (FORALL (a: TD_Fmla, t: Time): Alw(a) => Alw(Dist(a, t)))

Rule? (skolem!)
Skolemizing, this simplifies to:
theorem_viii :
  |-------
{1}    Alw(a!1) => Alw(Dist(a!1, t!1))

Rule? (flatten)
Applying disjunctive simplification to flatten sequent, this simplifies to:
theorem_viii :
{-1}    Alw(a!1)
  |-------
{1}    Alw(Dist(a!1, t!1))

Rule? (expand "Dist")
Expanding the definition of Dist, this simplifies to:
theorem_viii :
[-1]    Alw(a!1)
  |-------
{1}    Alw(LAMBDA (t: Time): a!1(t + t!1))

Rule? (expand "Alw")
Expanding the definition of Alw, this simplifies to:
theorem_viii :
{-1}    FORALL (tt: Time): a!1(tt)
  |-------
{1}    FORALL (tt: Time): a!1(tt + t!1)

Rule? (skolem!)
Skolemizing, this simplifies to:
theorem_viii :
[-1]    FORALL (tt: Time): a!1(tt)
  |-------
{1}    a!1(tt!1 + t!1)

Rule? (inst?)
Found substitution: tt gets tt!1 + t!1,
Using template: a!1(tt)
Instantiating quantified variables,
Q.E.D.
```

5.18 theorem viii.rev.

The reverse part of theorem viii, as I said, is not so easy to prove. Let's first prove a lemma.

```
inverse_add_time :
  |-------
{1}    (FORALL (t1: Time): -t1 + t1 = 0)
Rule? (grind)
Trying repeated skolemization, instantiation, and if-lifting,
Q.E.D.
```

And let's then apply it in the proof of the theorem. You'll see that the steps before the use of the lemma are re-run, this is because I tried to prove the theorem and did not manage to do it, I saved the uncomplete proof, I introduced and proved the lemma, and then I came back to complete the theorem proof.

```
theorem_viii_rev :
  |-------
{1}    (FORALL (a: TD_Fmla, t: Time): Alw(Dist(a, t)) => Alw(a))

Rerunning step: (SKOLEM!)
Skolemizing, this simplifies to:
theorem_viii_rev :
  |-------
{1}    Alw(Dist(a!1, t!1)) => Alw(a!1)

Rerunning step: (EXPAND "Dist")
Expanding the definition of Dist, this simplifies to:
theorem_viii_rev :
  |-------
{1}    (Alw(LAMBDA (t: Time): a!1(t + t!1)) => Alw(a!1))
Rerunning step: (EXPAND "Alw")
Expanding the definition of Alw, this simplifies to:
theorem_viii_rev :
  |-------
{1}    ((FORALL (tt: Time): a!1(tt + t!1)) => (FORALL (tt: Time): a!1(tt)))
Rerunning step: (SKOSIMP)
Skolemizing and flattening, this simplifies to:
theorem_viii_rev :
{-1}    FORALL (tt: Time): a!1(tt + t!1)
  |-------
{1}    FORALL (tt: Time): a!1(tt)
Rerunning step: (SKOLEM!)
Skolemizing, this simplifies to:
theorem_viii_rev :
[-1]    FORALL (tt: Time): a!1(tt + t!1)
  |-------
{1}    a!1(tt!1)
Rerunning step: (INSTANTIATE -1 "tt!1 - t!1")
Instantiating the top quantifier in -1 with the terms: tt!1 - t!1, this simplifies to:
theorem_viii_rev :
{-1}    a!1(tt!1 - t!1 + t!1)
  |-------
[1]    a!1(tt!1)
```

```
Rule? (lemma "inverse_add_time" ("t1" "t!1"))
Applying inverse_add_time where t1 gets t!1, this simplifies to:
theorem_viii_rev :
{-1}    -t!1 + t!1 = 0
[-2]    a!1(tt!1 - t!1 + t!1)
  |-------
[1]    a!1(tt!1)

Rule? (replace -2)
Replacing using formula -2, this simplifies to:
theorem_viii_rev :
[-1]    -t!1 + t!1 = 0
[-2]    a!1(tt!1 - t!1 + t!1)
  |-------
{1}    TRUE

which is trivially true.
Q.E.D.
```

5.19 theorem ix.

```
theorem_ix :
  |-------
{1}    (FORALL (a: TD_Fmla, p: nonnegtime):
         NOW(a AND LV(Alw(a IMPLIES Futr(a, p)))
                 IMPLIES (FA! (k: nat): Futr(a, k * p))))

Rerunning step: (EXPAND "NOW")
Expanding the definition of NOW, this simplifies to:
theorem_ix :
  |-------
{1}    (FORALL (a: TD_Fmla, p: nonnegtime):
         (a AND LV(Alw(a IMPLIES Futr(a, p)))
              IMPLIES (FA! (k: nat): Futr(a, k * p)))(0))

Rerunning step: (EXPAND "LV")
Expanding the definition of LV, this simplifies to:
theorem_ix :
  |-------
{1}    (FORALL (a: TD_Fmla, p: nonnegtime):
         ((a AND (LAMBDA (t: Time): Alw(a IMPLIES Futr(a, p))))
              IMPLIES (FA! (k: nat): Futr(a, k * p)))(0))

Rerunning step: (SKOLEM!)
Skolemizing, this simplifies to:
theorem_ix :
  |-------
{1}    ((a!1 AND (LAMBDA (t: Time): Alw(a!1 IMPLIES Futr(a!1, p!1))))
           IMPLIES (FA! (k: nat): Futr(a!1, k * p!1)))(0)
```

```
Rerunning step: (EXPAND "Alw")
Expanding the definition of Alw, this simplifies to:
theorem_ix :
  |-------
{1}    ((a!1
            AND
          (LAMBDA (t: Time):
             FORALL (tt: Time): (a!1 IMPLIES Futr(a!1, p!1))(tt)))
           IMPLIES (FA! (k: nat): Futr(a!1, k * p!1)))(0)

Rerunning step: (EXPAND "Futr")
Expanding the definition of Futr, this simplifies to:
theorem_ix :
  |-------
{1}    ((a!1
            AND
          (LAMBDA (t: Time):
             FORALL (tt: Time):
               (a!1 IMPLIES (LAMBDA (t: Time): a!1(t + p!1)))(tt)))
           IMPLIES (FA! (k: nat): LAMBDA (t: Time): a!1(t + k * p!1)))(0)

Rerunning step: (EXPAND "FA")
Expanding the definition of FA, this simplifies to:
theorem_ix :
  |-------
{1}    ((a!1
            AND
          (LAMBDA (t: Time):
             FORALL (tt: Time):
               (a!1 IMPLIES (LAMBDA (t: Time): a!1(t + p!1)))(tt)))
           IMPLIES (LAMBDA (t: Time): FORALL (x: nat): a!1(t + x * p!1)))(0)

Rerunning step: (EXPAND "IMPLIES")
Expanding the definition of IMPLIES, this simplifies to:
theorem_ix :
  |-------
{1}    (a!1
           AND
         (LAMBDA (t: Time):
            FORALL (tt: Time): a!1(tt) IMPLIES a!1(tt + p!1)))(0)
          IMPLIES (FORALL (x: nat): a!1(x * p!1))

Rerunning step: (FLATTEN)
Applying disjunctive simplification to flatten sequent, this simplifies to:
theorem_ix :
{-1}    (a!1
           AND
         (LAMBDA (t: Time):
            FORALL (tt: Time): a!1(tt) IMPLIES a!1(tt + p!1)))(0)
```

```
  |-------
{1}    FORALL (x: nat): a!1(x * p!1)

Rerunning step: (SKOLEM!)
Skolemizing, this simplifies to:
theorem_ix :
[-1]    (a!1
           AND
         (LAMBDA (t: Time):
            FORALL (tt: Time): a!1(tt) IMPLIES a!1(tt + p!1)))(0)
  |-------
{1}    a!1(x!1 * p!1)

Rerunning step: (INDUCT-AND-REWRITE! "tt")
By induction on tt and rewriting, this simplifies to:
theorem_ix :
[-1]    (a!1
           AND
         (LAMBDA (t: Time):
            FORALL (tt: Time): a!1(tt) IMPLIES a!1(tt + p!1)))(0)
  |-------
[1]    a!1(x!1 * p!1)

Rerunning step: (GROUND)
AND rewrites
   (a!1
       AND
     (LAMBDA (t: Time): FORALL (tt: Time): a!1(tt) IMPLIES a!1(tt + p!1)))(0)
  to  a!1(0) AND (FORALL (tt: Time): a!1(tt) IMPLIES a!1(tt + p!1))
Applying propositional simplification and decision procedures, this simplifies to:
theorem_ix :

{-1}    a!1(0) AND (FORALL (tt: Time): a!1(tt) IMPLIES a!1(tt + p!1))
  |-------
[1]    a!1(x!1 * p!1)

Rerunning step: (FLATTEN)
Applying disjunctive simplification to flatten sequent, this simplifies to:
theorem_ix :
{-1}    a!1(0)
{-2}    FORALL (tt: Time): a!1(tt) IMPLIES a!1(tt + p!1)
  |-------
[1]    a!1(x!1 * p!1)

Rerunning step: (LEMMA "NAT_induction")
Applying NAT_induction this simplifies to:
theorem_ix :
{-1}    (FORALL (p: pred[nat]):
         (FORALL (j: nat): (FORALL (k: nat): k < j IMPLIES p(k)) IMPLIES p(j))
             IMPLIES (FORALL (i: nat): p(i)))
```

```
[-2]    a!1(0)
[-3]    FORALL (tt: Time): a!1(tt) IMPLIES a!1(tt + p!1)
  |-------
[1]    a!1(x!1 * p!1)

Rerunning step: (INST - "(LAMBDA (n : nat): a!1(n * p!1))")
Instantiating the top quantifier in - with the terms:
 (LAMBDA (n : nat): a!1(n * p!1)), this simplifies to:
theorem_ix :
{-1}    (FORALL (j: nat):
         (FORALL (k: nat): k < j IMPLIES (LAMBDA (n: nat): a!1(n * p!1))(k))
             IMPLIES (LAMBDA (n: nat): a!1(n * p!1))(j))
          IMPLIES (FORALL (i: nat): (LAMBDA (n: nat): a!1(n * p!1))(i))
[-2]    a!1(0)
[-3]    FORALL (tt: Time): a!1(tt) IMPLIES a!1(tt + p!1)
  |-------
[1]    a!1(x!1 * p!1)

Rerunning step: (GROUND)
Applying propositional simplification and decision procedures,
this yields  2 subgoals:
theorem_ix.1 :
{-1}    (FORALL (i: nat): a!1(i * p!1))
[-2]    a!1(0)
[-3]    FORALL (tt: Time): a!1(tt) IMPLIES a!1(tt + p!1)
  |-------
[1]    a!1(x!1 * p!1)

Rerunning step: (GRIND)
Trying repeated skolemization, instantiation, and if-lifting,

This completes the proof of theorem_ix.1.

theorem_ix.2 :
[-1]    a!1(0)
[-2]    FORALL (tt: Time): a!1(tt) IMPLIES a!1(tt + p!1)
  |-------
{1}    (FORALL (j: nat):
         (FORALL (k: nat): k < j IMPLIES a!1(k * p!1)) IMPLIES a!1(j * p!1))
[2]    a!1(x!1 * p!1)

Rule? (quit)
```

5.20 theorem ix.bis.

```
theorem_ix_bis :
  |-------
{1}    (FORALL (a: TD_Fmla, t: Time):
         Alw((a AND LV(Alw(a IMPLIES Futr(a, t))))
                 IMPLIES FA! (k: nat): Futr(a, k * t)))
```

```
Rule? (skosimp)
Skolemizing and flattening, this simplifies to:
theorem_ix_bis :
  |-------
{1}    Alw((a!1 AND LV(Alw(a!1 IMPLIES Futr(a!1, t!1))))
              IMPLIES FA! (k: nat): Futr(a!1, k * t!1))

Rule? (expand "Alw")
Expanding the definition of Alw, this simplifies to:
theorem_ix_bis :
  |-------
{1}    FORALL (tt: Time):
        ((a!1 AND LV(FORALL (tt: Time): (a!1 IMPLIES Futr(a!1, t!1))(tt)))
             IMPLIES FA! (k: nat): Futr(a!1, k * t!1))(tt)

Rule? (skosimp)
Skolemizing and flattening, this simplifies to:
theorem_ix_bis :
  |-------
{1}    ((a!1 AND LV(FORALL (tt: Time): (a!1 IMPLIES Futr(a!1, t!1))(tt)))
           IMPLIES FA! (k: nat): Futr(a!1, k * t!1))(tt!1)

Rule? (expand "LV")
Expanding the definition of LV, this simplifies to:
theorem_ix_bis :
  |-------
{1}    ((a!1
            AND
          (LAMBDA (t: Time):
             FORALL (tt: Time): (a!1 IMPLIES Futr(a!1, t!1))(tt)))
           IMPLIES FA! (k: nat): Futr(a!1, k * t!1))(tt!1)

Rule? (expand "Futr")
Expanding the definition of Futr, this simplifies to:
theorem_ix_bis :
  |-------
{1}    ((a!1
            AND
          (LAMBDA (t: Time):
             FORALL (tt: Time):
               (a!1 IMPLIES (LAMBDA (t: Time): a!1(t + t!1)))(tt)))
           IMPLIES FA! (k: nat): LAMBDA (t: Time): a!1(t + k * t!1))(tt!1)

Rule? (expand "FA")
Expanding the definition of FA, this simplifies to:
theorem_ix_bis :
  |-------
{1}    ((a!1
            AND
```

```
          (LAMBDA (t: Time):
             FORALL (tt: Time):
                (a!1 IMPLIES (LAMBDA (t: Time): a!1(t + t!1)))(tt)))
           IMPLIES (LAMBDA (t: Time): FORALL (x: nat): a!1(t + x * t!1)))(tt!1)

Rule? (expand "IMPLIES")
Expanding the definition of IMPLIES, this simplifies to:
theorem_ix_bis :
  |-------
{1}    (a!1
           AND
         (LAMBDA (t: Time):
            FORALL (tt: Time): a!1(tt) IMPLIES a!1(tt + t!1)))(tt!1)
          IMPLIES (FORALL (x: nat): a!1(x * t!1 + tt!1))

Rule? (expand "AND")
Expanding the definition of AND, this simplifies to:
theorem_ix_bis :
  |-------
{1}    ((a!1(tt!1) AND (FORALL (tt: Time): a!1(tt) IMPLIES a!1(tt + t!1)))
           IMPLIES (FORALL (x: nat): a!1(x * t!1 + tt!1)))

Rule? (flatten)
Applying disjunctive simplification to flatten sequent, this simplifies to:
theorem_ix_bis :
{-1}    a!1(tt!1)
{-2}    FORALL (tt: Time): a!1(tt) IMPLIES a!1(tt + t!1)
  |-------
{1}    FORALL (x: nat): a!1(x * t!1 + tt!1)

Rule? (induct-and-rewrite! "tt")
By induction on tt and rewriting, this simplifies to:
theorem_ix_bis :
[-1]    a!1(tt!1)
[-2]    FORALL (tt: Time): a!1(tt) IMPLIES a!1(tt + t!1)
  |-------
[1]    FORALL (x: nat): a!1(x * t!1 + tt!1)

Rule? (quit)
```

5.21 theorem x.

A 99 percent completely proven theorem, a TCC remains out...

```
theorem_x :
  |-------
{1}    (FORALL (a: TD_Fmla, p: nonnegtime):
         NOW((a AND LV(Alw(a IMPLIES Futr(a, p))))
                 IMPLIES AlwF_i(WithinF_ii(a, p))))
```

```
Rerunning step: (EXPAND "NOW")
Expanding the definition of NOW, this simplifies to:
theorem_x :
  |-------
{1}    (FORALL (a: TD_Fmla, p: nonnegtime):
         ((a AND LV(Alw(a IMPLIES Futr(a, p))))
              IMPLIES AlwF_i(WithinF_ii(a, p)))(0))

Rerunning step: (EXPAND "LV")
Expanding the definition of LV, this simplifies to:
theorem_x :
  |-------
{1}    (FORALL (a: TD_Fmla, p: nonnegtime):
         ((a AND (LAMBDA (t: Time): Alw(a IMPLIES Futr(a, p))))
              IMPLIES AlwF_i(WithinF_ii(a, p)))(0))

Rerunning step: (EXPAND "Alw")
Expanding the definition of Alw, this simplifies to:
theorem_x :
  |-------
{1}    (FORALL (a: TD_Fmla, p: nonnegtime):
         ((a  AND
             (LAMBDA (t: Time):
                FORALL (tt: Time): (a IMPLIES Futr(a, p))(tt)))
              IMPLIES AlwF_i(WithinF_ii(a, p)))(0))

Rerunning step: (EXPAND "AlwF_i")
Expanding the definition of AlwF_i, this simplifies to:
theorem_x :
  |-------
{1}    (FORALL (a: TD_Fmla, p: nonnegtime):
         ((a  AND
             (LAMBDA (t: Time):
                FORALL (tt: Time): (a IMPLIES Futr(a, p))(tt)))
              IMPLIES
            (LAMBDA (t: Time):
               FORALL (nnt: nonnegtime): WithinF_ii(a, p)(nnt + t)))(0))

Rerunning step: (EXPAND "WithinF_ii")
Expanding the definition of WithinF_ii, this simplifies to:
theorem_x :
  |-------
{1}    (FORALL (a: TD_Fmla, p: nonnegtime):
         ((a  AND
             (LAMBDA (t: Time):
                FORALL (tt: Time): (a IMPLIES Futr(a, p))(tt)))
              IMPLIES
            (LAMBDA (t: Time):
               FORALL (nnt: nonnegtime):
                 EXISTS (it: {it: Time | 0 <= it AND it <= p}):
```

```
                 a(it + nnt + t)))(0))

Rerunning step: (SKOSIMP)
Skolemizing and flattening, this simplifies to:
theorem_x :
  |-------
{1}    ((a!1 AND
          (LAMBDA (t: Time):
             FORALL (tt: Time): (a!1 IMPLIES Futr(a!1, p!1))(tt)))
           IMPLIES
         (LAMBDA (t: Time):
            FORALL (nnt: nonnegtime):
              EXISTS (it: {it: Time | 0 <= it AND it <= p!1}):
                a!1(it + nnt + t)))(0)

Rerunning step: (EXPAND "AND")
Expanding the definition of AND, this simplifies to:
theorem_x :
  |-------
{1}    ((LAMBDA (t: Time):
          a!1(t) AND (FORALL (tt: Time): (a!1 IMPLIES Futr(a!1, p!1))(tt)))
           IMPLIES
         (LAMBDA (t: Time):
            FORALL (nnt: nonnegtime):
              EXISTS (it: {it: Time | 0 <= it AND it <= p!1}):
                a!1(it + nnt + t)))(0)

Rerunning step: (EXPAND "IMPLIES")
Expanding the definition of IMPLIES, this simplifies to:
theorem_x :
  |-------
{1}    (a!1(0) AND (FORALL (tt: Time): a!1(tt) IMPLIES Futr(a!1, p!1)(tt)))
          IMPLIES
        (FORALL (nnt: nonnegtime):
           EXISTS (it: {it: Time | 0 <= it AND it <= p!1}): a!1(it + nnt))

Rerunning step: (FLATTEN)
Applying disjunctive simplification to flatten sequent, this simplifies to:
theorem_x :
{-1}    a!1(0)
{-2}    FORALL (tt: Time): a!1(tt) IMPLIES Futr(a!1, p!1)(tt)
  |-------
{1}    FORALL (nnt: nonnegtime):
        EXISTS (it: {it: Time | 0 <= it AND it <= p!1}): a!1(it + nnt)

Rerunning step: (EXPAND "Futr")
Expanding the definition of Futr, this simplifies to:
theorem_x :
[-1]    a!1(0)
{-2}    FORALL (tt: Time): (a!1(tt) IMPLIES a!1(tt + p!1))
```

```
  |-------
[1]    FORALL (nnt: nonnegtime):
        EXISTS (it: {it: Time | 0 <= it AND it <= p!1}): a!1(it + nnt)

Rerunning step: (SKOSIMP)
Skolemizing and flattening, this simplifies to:
theorem_x :
[-1]    a!1(0)
[-2]    FORALL (tt: Time): (a!1(tt) IMPLIES a!1(tt + p!1))
  |-------
{1}    EXISTS (it: {it: Time | 0 <= it AND it <= p!1}): a!1(it + nnt!1)

Rerunning step: (INST -2 "0")
Instantiating the top quantifier in -2 with the terms: 0, this simplifies to:
theorem_x :
[-1]    a!1(0)
{-2}    (a!1(0) IMPLIES a!1(0 + p!1))
  |-------
[1]    EXISTS (it: {it: Time | 0 <= it AND it <= p!1}): a!1(it + nnt!1)

Rerunning step: (BDDSIMP)
Applying bddsimp, this simplifies to:
theorem_x :
[-1]    a!1(0)
{-2}    a!1(0 + p!1)
  |-------
[1]    EXISTS (it: {it: Time | 0 <= it AND it <= p!1}): a!1(it + nnt!1)

Rerunning step: (INST 1 "-nnt!1 + p!1")
Instantiating the top quantifier in 1 with the terms:
 -nnt!1 + p!1, this yields  2 subgoals:
theorem_x.1 :
[-1]    a!1(0)
[-2]    a!1(0 + p!1)
  |-------
{1}    a!1(-nnt!1 + p!1 + nnt!1)

Rule? (replace 1)
Replacing using formula 1, this simplifies to:
theorem_x.1 :
[-1]    -nnt!1 + p!1 + nnt!1 = p!1
[-2]    a!1(0)
{-3}    FALSE
  |-------
[1]    a!1(-nnt!1 + p!1 + nnt!1)

which is trivially true.
This completes the proof of theorem_x.1.
theorem_x.2 (TCC):
[-1]    a!1(0)
```

```
[-2]    a!1(0 + p!1)
  |-------
{1}    0 <= -nnt!1 + p!1

Rule?
```

5.22 theorem xi.

This has been a quite difficult proof for me, but successful!

```
theorem_xi :
  |-------
{1}    (FORALL (a: TD_Fmla, t: Time):
         NOW(LAMBDA (x: Time):
               Futr(a, t)(x) => Lasts_ii(WithinF_ii(a, t), t)(x)))
Rule? (skolem!)
Skolemizing, this simplifies to:
theorem_xi :
  |-------
{1}    NOW(LAMBDA (x: Time):
            Futr(a!1, t!1)(x) => Lasts_ii(WithinF_ii(a!1, t!1), t!1)(x))

Rule? (expand "NOW")
Expanding the definition of NOW, this simplifies to:
theorem_xi :
  |-------
{1}    Futr(a!1, t!1)(0) => Lasts_ii(WithinF_ii(a!1, t!1), t!1)(0)

Rule? (flatten)
Applying disjunctive simplification to flatten sequent, this simplifies to:
theorem_xi :
{-1}    Futr(a!1, t!1)(0)
  |-------
{1}    Lasts_ii(WithinF_ii(a!1, t!1), t!1)(0)

Rule? (expand "Futr")
Expanding the definition of Futr, this simplifies to:
theorem_xi :
{-1}    a!1(t!1)
  |-------
[1]    Lasts_ii(WithinF_ii(a!1, t!1), t!1)(0)

Rule? (expand "Lasts_ii")
Expanding the definition of Lasts_ii, this simplifies to:
theorem_xi :
[-1]    a!1(t!1)
  |-------
{1}    FORALL (it: {it: Time | 0 <= it AND it <= t!1}): WithinF_ii(a!1, t!1)(it)

Rule? (expand "WithinF_ii")
```

```
Expanding the definition of WithinF_ii, this simplifies to:
theorem_xi :
[-1]    a!1(t!1)
  |-------
{1}    FORALL (it: {it: Time | 0 <= it AND it <= t!1}):
        EXISTS (it_69: {it: Time | 0 <= it AND it <= t!1}): a!1(it + it_69)

Rule? (skolem!)
Skolemizing, this simplifies to:
theorem_xi :
[-1]    a!1(t!1)
  |-------
{1}    EXISTS (it_69: {it: Time | 0 <= it AND it <= t!1}): a!1(it!1 + it_69)

Rule? (inst 1 "t!1 - it!1 -it_69")
No change on: (INST 1 "t!1 - it!1 -it_69")
theorem_xi :

[-1]    a!1(t!1)
  |-------
{1}    EXISTS (it_69: {it: Time | 0 <= it AND it <= t!1}): a!1(it!1 + it_69)

Rule? (instantiate 1 "t!1 - it!1")
Instantiating the top quantifier in 1 with the terms: t!1 - it!1, this simplifies to:
theorem_xi :
[-1]    a!1(t!1)
  |-------
{1}    a!1(it!1 + (t!1 - it!1))

Rule? (bddsimp)
No change on: (BDDSIMP)
theorem_xi :
[-1]    a!1(t!1)
  |-------
{1}    a!1(it!1 + (t!1 - it!1))

Rule? (ground)
Applying propositional simplification and decision procedures,
Q.E.D.
```

5.23 theorem xii.

I nearly managed to prove this theorem. There's only a final TCC to prove I cannot solve. I'm sure you'll be able to dispose of if without effort.

```
theorem_xii :
  |-------
{1}    (FORALL (a: TD_Fmla, p: nonnegtime, q: nonnegtime):
         NOW((a AND LV(Alw(a IMPLIES Futr(WithinF_ii(a, p), q))))
                IMPLIES AlwF_e(WithinF_ii(a, p + q))))
```

```
Rerunning step: (EXPAND "NOW")
Expanding the definition of NOW, this simplifies to:
theorem_xii :
  |-------
{1}    (FORALL (a: TD_Fmla, p: nonnegtime, q: nonnegtime):
         ((a AND LV(Alw(a IMPLIES Futr(WithinF_ii(a, p), q))))
              IMPLIES AlwF_e(WithinF_ii(a, p + q)))(0))

Rerunning step: (EXPAND "LV")
Expanding the definition of LV, this simplifies to:
theorem_xii :
  |-------
{1}    (FORALL (a: TD_Fmla, p: nonnegtime, q: nonnegtime):
         ((a AND (LAMBDA (t: Time): Alw(a IMPLIES Futr(WithinF_ii(a, p), q))))
              IMPLIES AlwF_e(WithinF_ii(a, p + q)))(0))

Rerunning step: (EXPAND "Alw")
Expanding the definition of Alw, this simplifies to:
theorem_xii :
  |-------
{1}    (FORALL (a: TD_Fmla, p: nonnegtime, q: nonnegtime):
         ((a
               AND
             (LAMBDA (t: Time):
                FORALL (tt: Time): (a IMPLIES Futr(WithinF_ii(a, p), q))(tt)))
              IMPLIES AlwF_e(WithinF_ii(a, p + q)))(0))

Rerunning step: (EXPAND "AlwF_e")
Expanding the definition of AlwF_e, this simplifies to:
theorem_xii :
  |-------
{1}    (FORALL (a: TD_Fmla, p: nonnegtime, q: nonnegtime):
         ((a
               AND
             (LAMBDA (t: Time):
                FORALL (tt: Time): (a IMPLIES Futr(WithinF_ii(a, p), q))(tt)))
              IMPLIES
            (LAMBDA (t: Time):
               FORALL (pt: postime): WithinF_ii(a, p + q)(pt + t)))(0))

Rerunning step: (EXPAND "WithinF_ii")
Expanding the definition of WithinF_ii, this simplifies to:
theorem_xii :
  |-------
{1}    (FORALL (a: TD_Fmla, p: nonnegtime, q: nonnegtime):
         ((a
               AND
             (LAMBDA (t: Time):
                FORALL (tt: Time):
```

```
                (a
                     IMPLIES
                   Futr(LAMBDA (t: Time):
                          EXISTS (it: {it: Time | 0 <= it AND it <= p}):
                            a(it + t),
                        q))(tt)))
            IMPLIES
          (LAMBDA (t: Time):
             FORALL (pt: postime):
               EXISTS (it: {it: Time | 0 <= it AND it <= p + q}):
                 a(it + pt + t)))(0))

Rerunning step: (SKOSIMP)
Skolemizing and flattening, this simplifies to:
theorem_xii :
  |-------
{1}    ((a!1
           AND
         (LAMBDA (t: Time):
            FORALL (tt: Time):
              (a!1
                   IMPLIES
                 Futr(LAMBDA (t: Time):
                        EXISTS (it: {it: Time | 0 <= it AND it <= p!1}):
                          a!1(it + t),
                      q!1))(tt)))
          IMPLIES
        (LAMBDA (t: Time):
           FORALL (pt: postime):
             EXISTS (it: {it: Time | 0 <= it AND it <= p!1 + q!1}):
               a!1(it + pt + t)))(0)

Rerunning step: (EXPAND "IMPLIES")
Expanding the definition of IMPLIES, this simplifies to:
theorem_xii :
  |-------
{1}    (a!1
          AND
        (LAMBDA (t: Time):
           FORALL (tt: Time):
             a!1(tt)
                 IMPLIES
               Futr(LAMBDA (t: Time):
                      EXISTS (it: {it: Time | 0 <= it AND it <= p!1}):
                        a!1(it + t),
                    q!1)(tt)))(0)
         IMPLIES
       (FORALL (pt: postime):
          EXISTS (it: {it: Time | 0 <= it AND it <= p!1 + q!1}): a!1(it + pt))
```

```
Rerunning step: (EXPAND "Futr")
Expanding the definition of Futr, this simplifies to:
theorem_xii :
  |-------
{1}    ((a!1
            AND
          (LAMBDA (t: Time):
             FORALL (tt: Time):
               (a!1(tt)
                    IMPLIES
                  (EXISTS (it: {it: Time | 0 <= it AND it <= p!1}):
                     a!1(it + tt + q!1))))))(0)
           IMPLIES
         (FORALL (pt: postime):
            EXISTS (it: {it: Time | 0 <= it AND it <= p!1 + q!1}):
              a!1(it + pt)))

Rerunning step: (SKOSIMP)
Skolemizing and flattening, this simplifies to:
theorem_xii :

{-1}    (a!1
           AND
         (LAMBDA (t: Time):
            FORALL (tt: Time):
              (a!1(tt)
                   IMPLIES
                 (EXISTS (it: {it: Time | 0 <= it AND it <= p!1}):
                    a!1(it + tt + q!1))))))(0)
  |-------
{1}    FORALL (pt: postime):
        EXISTS (it: {it: Time | 0 <= it AND it <= p!1 + q!1}): a!1(it + pt)

Rerunning step: (SKOLEM!)
Skolemizing, this simplifies to:
theorem_xii :
[-1]    (a!1
           AND
         (LAMBDA (t: Time):
            FORALL (tt: Time):
              (a!1(tt)
                   IMPLIES
                 (EXISTS (it: {it: Time | 0 <= it AND it <= p!1}):
                    a!1(it + tt + q!1))))))(0)
  |-------
{1}    EXISTS (it: {it: Time | 0 <= it AND it <= p!1 + q!1}): a!1(it + pt!1)

Rerunning step: (EXPAND "AND")
Expanding the definition of AND, this simplifies to:
theorem_xii :
```

```
{-1}    a!1(0)
          AND
        (FORALL (tt: Time):
           (a!1(tt)
                IMPLIES
              (EXISTS (it: {it: Time | 0 <= it AND it <= p!1}):
                 a!1(it + tt + q!1))))
  |-------
[1]    EXISTS (it: {it: Time | 0 <= it AND it <= p!1 + q!1}): a!1(it + pt!1)

Rerunning step: (SKOSIMP)
Skolemizing and flattening, this simplifies to:
theorem_xii :
{-1}    a!1(0)
{-2}    FORALL (tt: Time):
        (a!1(tt)
             IMPLIES
           (EXISTS (it: {it: Time | 0 <= it AND it <= p!1}):
              a!1(it + tt + q!1)))
  |-------
[1]    EXISTS (it: {it: Time | 0 <= it AND it <= p!1 + q!1}): a!1(it + pt!1)

Rerunning step: (INST -2 "0")
Instantiating the top quantifier in -2 with the terms: 0, this simplifies to:
theorem_xii :
[-1]    a!1(0)
{-2}    (a!1(0)
           IMPLIES
         (EXISTS (it: {it: Time | 0 <= it AND it <= p!1}): a!1(it + 0 + q!1)))
  |-------
[1]    EXISTS (it: {it: Time | 0 <= it AND it <= p!1 + q!1}): a!1(it + pt!1)

Rerunning step: (BDDSIMP)
Applying bddsimp, this simplifies to:
theorem_xii :
[-1]    a!1(0)
{-2}    EXISTS (it: {it: Time | 0 <= it AND it <= p!1}): a!1(it + 0 + q!1)
  |-------
[1]    EXISTS (it: {it: Time | 0 <= it AND it <= p!1 + q!1}): a!1(it + pt!1)

Rerunning step: (SKOSIMP)
Skolemizing and flattening, this simplifies to:
theorem_xii :
[-1]    a!1(0)
{-2}    a!1(it!1 + 0 + q!1)
  |-------
[1]    EXISTS (it: {it: Time | 0 <= it AND it <= p!1 + q!1}): a!1(it + pt!1)

Rerunning step: (INST 1 "it!1 -pt!1 + q!1")
Instantiating the top quantifier in 1 with the terms: it!1 -pt!1 + q!1,
```

```
this yields  2 subgoals:
theorem_xii.1 :
[-1]    a!1(0)
[-2]    a!1(it!1 + 0 + q!1)
  |-------
{1}    a!1(it!1 - pt!1 + q!1 + pt!1)

Rerunning step: (LEMMA "inverse_add_time_const2")
Applying inverse_add_time_const2, this simplifies to:
theorem_xii.1 :
{-1}    (FORALL (t1: Time, t2: Time): -t1 + t2 + t1 = t2)
[-2]    a!1(0)
[-3]    a!1(it!1 + 0 + q!1)
  |-------
[1]    a!1(it!1 - pt!1 + q!1 + pt!1)

Rerunning step: (INST -1 "pt!1" "q!1")
Instantiating the top quantifier in -1 with the terms: pt!1, q!1, this simplifies to:
theorem_xii.1 :
{-1}    -pt!1 + q!1 + pt!1 = q!1
[-2]    a!1(0)
[-3]    a!1(it!1 + 0 + q!1)
  |-------
[1]    a!1(it!1 - pt!1 + q!1 + pt!1)

Rerunning step: (REPLACE -1)
Replacing using formula -1, this simplifies to:
theorem_xii.1 :
[-1]    -pt!1 + q!1 + pt!1 = q!1
[-2]    a!1(0)
{-3}    a!1(it!1 + 0 + q!1)
  |-------
{1}    a!1(it!1 - pt!1 + q!1 + pt!1)

Rerunning step: (REPLACE 1)
Replacing using formula 1, this simplifies to:
theorem_xii.1 :
[-1]    -pt!1 + q!1 + pt!1 = q!1
[-2]    a!1(0)
{-3}    FALSE
  |-------
[1]    a!1(it!1 - pt!1 + q!1 + pt!1)

which is trivially true.

This completes the proof of theorem_xii.1.
theorem_xii.2 (TCC):
[-1]    a!1(0)
[-2]    a!1(it!1 + 0 + q!1)
  |-------
```

```
{1}    0 <= it!1 - pt!1 + q!1

Rule?
```

5.24 theorem xiv.

This is the log (with some cuts in the mosts boring parts) of the proof:

```
theorem_xiv :
  |-------
{1}    (FORALL (a: TD_Fmla, t: Time):
         NOW(WithinF_ee(a, t)
                 IMPLIES (FA! (d: Time): (LV(d > t) IMPLIES WithinF_ee(a, d)))))

Rule? (grind$)
Auto-rewriting with all the definitions relevant to conjecture,
this simplifies to:
theorem_xiv :
  |-------
[1]    (FORALL (a: TD_Fmla, t: Time):
         NOW(WithinF_ee(a, t)
                 IMPLIES (FA! (d: Time): (LV(d > t) IMPLIES WithinF_ee(a, d)))))

WithinF_ee rewrites WithinF_ee(a, t)(0)
  to  EXISTS (it: {it: Time | 0 < it AND it < t}): a(it)
LV rewrites LV(x > t)(0)
  to  x > t
WithinF_ee rewrites WithinF_ee(a, x)(0)
  to  EXISTS (it: {it: Time | 0 < it AND it < x}): a(it)
IMPLIES rewrites (LV(x > t) IMPLIES WithinF_ee(a, x))(0)
  to  x > t IMPLIES (EXISTS (it: {it: Time | 0 < it AND it < x}): a(it))
FA rewrites (FA! (d: Time): (LV(d > t) IMPLIES WithinF_ee(a, d)))(0)
  to  FORALL (x: Time):
        x > t IMPLIES (EXISTS (it: {it: Time | 0 < it AND it < x}): a(it))
IMPLIES rewrites
   (WithinF_ee(a, t) IMPLIES (FA! (d: Time): (LV(d > t) IMPLIES WithinF_ee(a, d))))(0)
  to  (EXISTS (it: {it: Time | 0 < it AND it < t}): a(it))
          IMPLIES
        (FORALL (x: Time):
           x > t IMPLIES (EXISTS (it: {it: Time | 0 < it AND it < x}): a(it)))
NOW rewrites
   NOW(WithinF_ee(a, t) IMPLIES (FA! (d: Time): (LV(d > t) IMPLIES WithinF_ee(a, d))))
  to  (EXISTS (it: {it: Time | 0 < it AND it < t}): a(it))
          IMPLIES
        (FORALL (x: Time):
           x > t IMPLIES (EXISTS (it: {it: Time | 0 < it AND it < x}): a(it)))
Simplifying, rewriting, and recording with decision procedures,
this simplifies to:
theorem_xiv :
  |-------
```

```
{1}    (FORALL (a: TD_Fmla, t: Time):
         (EXISTS (it: {it: Time | 0 < it AND it < t}): a(it))
             IMPLIES
           (FORALL (x: Time):
              x > t IMPLIES (EXISTS (it: {it: Time | 0 < it AND it < x}): a(it))))

Skolemizing (with typepred on new Skolem constants), this simplifies to:
theorem_xiv :
{-1}    real_pred(t!1)
  |-------
{1}    (EXISTS (it: {it: Time | 0 < it AND it < t!1}): a!1(it))
          IMPLIES
        (FORALL (x: Time):
           x > t!1 IMPLIES (EXISTS (it: {it: Time | 0 < it AND it < x}): a!1(it)))

Applying disjunctive simplification to flatten sequent, this simplifies to:
theorem_xiv :
[-1]    real_pred(t!1)
{-2}    EXISTS (it: {it: Time | 0 < it AND it < t!1}): a!1(it)
  |-------
{1}    FORALL (x: Time):
        x > t!1 IMPLIES (EXISTS (it: {it: Time | 0 < it AND it < x}): a!1(it))

Skolemizing (with typepred on new Skolem constants), this simplifies to:
theorem_xiv :
{-1}    0 < it!1
{-2}    it!1 < t!1
[-3]    real_pred(t!1)
{-4}    a!1(it!1)
  |-------
[1]    FORALL (x: Time):
        x > t!1 IMPLIES (EXISTS (it: {it: Time | 0 < it AND it < x}): a!1(it))

Skolemizing (with typepred on new Skolem constants), this simplifies to:
theorem_xiv :
{-1}    real_pred(x!1)
[-2]    0 < it!1
[-3]    it!1 < t!1
[-4]    real_pred(t!1)
[-5]    a!1(it!1)
  |-------
{1}    x!1 > t!1 IMPLIES (EXISTS (it: {it: Time | 0 < it AND it < x!1}): a!1(it))

Applying disjunctive simplification to flatten sequent,
this simplifies to:
theorem_xiv :
[-1]    real_pred(x!1)
[-2]    0 < it!1
[-3]    it!1 < t!1
[-4]    real_pred(t!1)
```

```
[-5]    a!1(it!1)
{-6}    x!1 > t!1
  |-------
{1}    EXISTS (it: {it: Time | 0 < it AND it < x!1}): a!1(it)

Found substitution: it gets it!1,
Using template: a!1(it)
Instantiating quantified variables,
Q.E.D.
```

5.25 theorem xv.

Theorem xv's proof has been very difficult to prove for me. I had to invent and prove the "inverse_add_time_const" lemma before:

```
inverse_add_time_const :
  |-------
{1}    (FORALL (t1: Time, t2: Time): t1 + t2 - t1 = t2)
Rerunning step: (GRIND)
Trying repeated skolemization, instantiation, and if-lifting,
Q.E.D.
```

And then use it to prove the main theorem:

```
theorem_xv :
  |-------
{1}    (FORALL (a: TD_Fmla, d: Time, t: Time):
         NOW(Futr(WithinF_ee(a, t), d) AND LV(t > 0)
                 IMPLIES FA! (s: Time):
                 ((LV((0 < s AND s < d))
                      IMPLIES Futr(WithinF_ee(a, t + s), d - s))))

Rule? (expand "Futr")
Expanding the definition of Futr, this simplifies to:
theorem_xv :
  |-------
{1}    (FORALL (a: TD_Fmla, d: Time, t: Time):
         NOW((((LAMBDA (t_85: Time): WithinF_ee(a, t)(d + t_85)) AND LV(t > 0))
                  IMPLIES FA! (s: Time):
                  ((LV((0 < s AND s < d))
                       IMPLIES
                     (LAMBDA (t_86: Time):
                        WithinF_ee(a, s + t)(d + t_86 - s))))))

Rule? (skosimp)
Skolemizing and flattening, this simplifies to:
theorem_xv :
  |-------
{1}    NOW((((LAMBDA (t_85: Time): WithinF_ee(a!1, t!1)(d!1 + t_85))
                AND LV(t!1 > 0))
               IMPLIES FA! (s: Time):
```

```
              ((LV((0 < s AND s < d!1)))
                   IMPLIES
                 (LAMBDA (t_86: Time):
                    WithinF_ee(a!1, s + t!1)(d!1 + t_86 - s)))))

Rule? (expand "NOW")
Expanding the definition of NOW, this simplifies to:
theorem_xv :
  |-------
{1}    (((LAMBDA (t_85: Time): WithinF_ee(a!1, t!1)(t_85 + d!1))
            AND LV(t!1 > 0))
           IMPLIES FA! (s: Time):
           ((LV((0 < s AND s < d!1)))
                IMPLIES
              (LAMBDA (t_86: Time):
                 WithinF_ee(a!1, s + t!1)(t_86 - s + d!1))))(0)

Rule? (expand "LV")
Expanding the definition of LV, this simplifies to:
theorem_xv :
  |-------
{1}    (((LAMBDA (t_85: Time): WithinF_ee(a!1, t!1)(t_85 + d!1))
            AND (LAMBDA (t: Time): t!1 > 0))
           IMPLIES FA! (s: Time):
           ((LAMBDA (t: Time): (0 < s AND s < d!1))
                IMPLIES
              (LAMBDA (t_86: Time):
                 WithinF_ee(a!1, s + t!1)(t_86 - s + d!1))))(0)

Rule? (expand "IMPLIES")
Expanding the definition of IMPLIES, this simplifies to:
theorem_xv :
  |-------
{1}    ((LAMBDA (t_85: Time): WithinF_ee(a!1, t!1)(t_85 + d!1))
           AND (LAMBDA (t: Time): t!1 > 0))(0)
          IMPLIES
        FA! (s: Time):
          (LAMBDA (t: Time):
             (0 < s AND s < d!1)
                 IMPLIES WithinF_ee(a!1, s + t!1)(t - s + d!1))(0)

Rule? (expand "FA")
Expanding the definition of FA, this simplifies to:
theorem_xv :
  |-------
{1}    (((LAMBDA (t_85: Time): WithinF_ee(a!1, t!1)(t_85 + d!1))
            AND (LAMBDA (t: Time): t!1 > 0))(0)
           IMPLIES
         (FORALL (x: Time):
            (0 < x AND x < d!1) IMPLIES WithinF_ee(a!1, x + t!1)(d!1 - x)))
```

```
Rule? (flatten)
Applying disjunctive simplification to flatten sequent, this simplifies to:
theorem_xv :
{-1}    ((LAMBDA (t_85: Time): WithinF_ee(a!1, t!1)(t_85 + d!1))
           AND (LAMBDA (t: Time): t!1 > 0))(0)
  |-------
{1}    FORALL (x: Time):
        (0 < x AND x < d!1) IMPLIES WithinF_ee(a!1, x + t!1)(d!1 - x)

Rule? (expand "AND")
Expanding the definition of AND, this simplifies to:
theorem_xv :
{-1}    WithinF_ee(a!1, t!1)(d!1) AND t!1 > 0
  |-------
[1]    FORALL (x: Time):
        (0 < x AND x < d!1) IMPLIES WithinF_ee(a!1, x + t!1)(d!1 - x)

Rule? (expand "WithinF_ee")
Expanding the definition of WithinF_ee, this simplifies to:
theorem_xv :
{-1}    ((EXISTS (it: {it: Time | 0 < it AND it < t!1}): a!1(it + d!1))
           AND t!1 > 0)
  |-------
{1}    FORALL (x: Time):
        ((0 < x AND x < d!1)
             IMPLIES
           (EXISTS (it: {it: Time | 0 < it AND it < x + t!1}):
              a!1(it - x + d!1)))

Rule? (skosimp!)
Ill-formed rule/strategy: (SKOSIMP!)
No change on: (SKIP)
theorem_xv :
{-1}    ((EXISTS (it: {it: Time | 0 < it AND it < t!1}): a!1(it + d!1))
           AND t!1 > 0)
  |-------
{1}    FORALL (x: Time):
        ((0 < x AND x < d!1)
             IMPLIES
           (EXISTS (it: {it: Time | 0 < it AND it < x + t!1}):
              a!1(it - x + d!1)))

Rule? (skosimp)
Skolemizing and flattening, this simplifies to:
theorem_xv :
{-1}    EXISTS (it: {it: Time | 0 < it AND it < t!1}): a!1(it + d!1)
{-2}    t!1 > 0
{-3}    0 < x!1
{-4}    x!1 < d!1
```

```
  |-------
{1}    EXISTS (it: {it: Time | 0 < it AND it < t!1 + x!1}): a!1(it - x!1 + d!1)

Rule? (skolem!)
Skolemizing, this simplifies to:
theorem_xv :
{-1}    a!1(it!1 + d!1)
[-2]    t!1 > 0
[-3]    0 < x!1
[-4]    x!1 < d!1
  |-------
[1]    EXISTS (it: {it: Time | 0 < it AND it < t!1 + x!1}): a!1(it - x!1 + d!1)

Rule? (instantiate 1 "x!1 + it!1")
Instantiating the top quantifier in 1 with the terms: x!1 + it!1,
this yields  2 subgoals:
theorem_xv.1 :
[-1]    a!1(it!1 + d!1)
[-2]    t!1 > 0
[-3]    0 < x!1
[-4]    x!1 < d!1
  |-------
{1}    a!1(x!1 + it!1 - x!1 + d!1)

Rule? (lemma "inverse_add_time_const" ("t1" "x!1" "t2" "it!1"))
Applying inverse_add_time_const where
  t1 gets x!1,
  t2 gets it!1,
this simplifies to:
theorem_xv.1 :
{-1}    x!1 + it!1 - x!1 = it!1
[-2]    a!1(it!1 + d!1)
[-3]    t!1 > 0
[-4]    0 < x!1
[-5]    x!1 < d!1
  |-------
[1]    a!1(x!1 + it!1 - x!1 + d!1)

Rule? (replace -1)
Replacing using formula -1, this simplifies to:
theorem_xv.1 :
[-1]    x!1 + it!1 - x!1 = it!1
{-2}    a!1(it!1 + d!1)
[-3]    t!1 > 0
[-4]    0 < x!1
[-5]    x!1 < d!1
  |-------
{1}    a!1(it!1 + d!1)

which is trivially true.
```

```
This completes the proof of theorem_xv.1.
theorem_xv.2 (TCC):
[-1]    a!1(it!1 + d!1)
[-2]    t!1 > 0
[-3]    0 < x!1
[-4]    x!1 < d!1
  |-------
{1}    0 < it!1 + x!1

Rule? (grind)
Trying repeated skolemization, instantiation, and if-lifting,

This completes the proof of theorem_xv.2.
Q.E.D.
```

6 Appendix a: PVS full file

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Politecnico di Milano - University of Illinois at Chicago %
%                                                           %
% Formal Methods exam - professor Mandrioli                 %
%                                                           %
% Homework by Daniele Paolo Scarpazza                       %
%                                                           %
%       Copyright (C) 2000, by Daniele Paolo Scarpazza      %
%       Copyright (C) portions 2000, by Angelo Gargantini   %
%       All rights reserved.                                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

trioExamples  : THEORY
BEGIN
  importing trio_std, trio_arith, trio_quantif[int]

  n             : VAR nat
  p,q           : VAR nonnegtime
  d,t,s,t1,t2   : VAR Time
  start_t,end_t : VAR Time

  dur           : VAR nonnegtime
  k             : VAR nat
  x             : var int
  a,b           : var TD_Fmla

  theorem_i: theorem
   Alw(Alw(a) IMPLIES Dist(a,t))

  theorem_i_bis: theorem
   Alw(a) IMPLIES NOW(Dist(a,t))
```

```
  theorem_i_ter: theorem
   NOW(Alw(a) IMPLIES Dist(a,t))

  theorem_ii: theorem
   Alw(Alw(a) IMPLIES a)

  theorem_ii_bis: theorem
   Alw(a) IMPLIES NOW(a)

  theorem_ii_ter: theorem
   NOW(Alw(a) IMPLIES a)

  theorem_iii: theorem
   Alw((Alw(a IMPLIES b) AND Dist(a,t)) IMPLIES Dist(b,t))

  theorem_iii_bis: theorem
   Alw(a IMPLIES b) AND NOW(Dist(a,t)) IMPLIES NOW(Dist(b,t))

  theorem_iv: THEOREM
   Alw( (    ( a IMPLIES Alw(a) )
         AND ( (NOT a) IMPLIES (Alw(NOT a)))
         AND ( Dist(a,t))) IMPLIES a)

  theorem_v: THEOREM
   Alw( Futr(a IMPLIES b, t) IMPLIES (Futr(a,t) IMPLIES Futr(b,t)) )

  theorem_v_bis: THEOREM
   NOW(Futr(a IMPLIES b, t) IMPLIES (Futr(a,t) IMPLIES Futr(b,t)) )

  theorem_vi_a: THEOREM
   NOW(Futr(Futr(a,t1),t2) => Futr(a, t1 + t2))

  theorem_vi_b_original: THEOREM
   NOW(Futr(Past(a,t1),t2) AND t1 > t2 IMPLIES Futr(a, t1 - t2))
%Fail; from page 11 of FMMX, it must be corrected

  theorem_vi_b_correct: THEOREM
   NOW(Futr(Past(a,t1),t2) AND t1 > t2 IMPLIES Futr(a, t2 - t1))

  theorem_vi_c_original: THEOREM
   NOW(Futr(Past(a,t1),t2) AND t1 < t2 IMPLIES Past(a, t2 - t1))
%Fail; from page 11 of FMMX, it must be corrected

  theorem_vi_c_correct: THEOREM
   NOW(Futr(Past(a,t1),t2) AND t1 < t2 IMPLIES Past(a, t1 - t2))

  theorem_vii_a: THEOREM
   NOW(Dist(FA! (x): a,t) IMPLIES FA!(x): Dist(a,t))
```

```
  theorem_vii_b: THEOREM
   NOW(Dist(EX! (x): a,t) IMPLIES EX!(x): Dist(a,t))

  theorem_vii_a_rev: THEOREM
   NOW(FA!(x): Dist(a,t) IMPLIES  Dist(FA! (x): a,t))


  theorem_vii_b_rev: THEOREM
   NOW( EX!(x): Dist(a,t)) IMPLIES NOW(Dist(EX! (x): a,t))

  theorem_viii: THEOREM
   Alw(a) => Alw(Dist(a,t))

  inverse_add_time : LEMMA
   -t1 + t1 = 0

  theorem_viii_rev: THEOREM
   Alw(Dist(a,t)) => Alw(a)

  theorem_ix: THEOREM
   NOW(a AND Alw(a IMPLIES Futr(a,p)) IMPLIES (FA! (k): Futr(a,k*p)))

  theorem_ix_bis: THEOREM
   Alw( (a AND Alw(a IMPLIES Futr(a,t))) IMPLIES FA! (k): Futr(a,k*t))

  inverse_add_time_const2:LEMMA
   -t1 + t2 + t1 = t2

  theorem_x: THEOREM
   NOW( (a AND Alw(a IMPLIES Futr(a,t))) IMPLIES AlwF_i(WithinF_ii(a,t)))

  theorem_xi: THEOREM
   NOW(Futr(a,t) => Lasts_ii(WithinF_ii(a,t),t))

  theorem_xii: THEOREM
   NOW( (a AND Alw(a IMPLIES Futr(WithinF_ii(a,p),q))) IMPLIES AlwF_e(WithinF_ii(a,p+q)))

  theorem_xiii: THEOREM
   NOW((FA! (n) : Futr(a AND Lasted_ii(a,t), t*n)) IMPLIES NOW(AlwF_e(a)))

  theorem_xiv: THEOREM
   NOW( WithinF_ee(a,t) IMPLIES (FA! (d): ( d > t IMPLIES WithinF_ee(a,d))))

  inverse_add_time_const:LEMMA
   t1 + t2 - t1 = t2

  theorem_xv: THEOREM
   NOW(Futr(WithinF_ee(a,t),d) AND t>0 IMPLIES FA! (s) : ((0<s AND s<d) IMPLIES Futr(WithinF_

END trioExamples
```

7 Appendix b: Proof file

```
(|trioExamples|
 (|theorem_i| "" (EXPAND "Alw")
  ((""(EXPAND "LV")
    ((""(EXPAND "IMPLIES")
      ((""(SKOLEM!)
        ((""(GROUND)
          ((""(EXPAND "Dist")
            ((""(SKOLEM!)
              ((""(FLATTEN)
                ((""(INSTANTIATE - "tt!1 + t!1")
                  ((""(PROPAX) NIL)))))))))))))))))))
 (|theorem_i_bis| "" (EXPAND "Alw")
  ((""(EXPAND "NOW")
    ((""(EXPAND "Dist")
      ((""(GROUND)
        ((""(SKOLEM!)
          ((""(FLATTEN)
            ((""(INSTANTIATE -1 "t!1")
              ((""(PROPAX) NIL)))))))))))))))
 (|theorem_i_ter| "" (SKOLEM!)
  ((""(EXPAND "NOW")
    ((""(EXPAND "LV")
      ((""(EXPAND "Alw")
        ((""(EXPAND "IMPLIES")
          ((""(FLATTEN)
            ((""(EXPAND "Dist")
              ((""(INSTANTIATE -1 "t!1")
                ((""(PROPAX) NIL)))))))))))))))))
 (|theorem_ii| "" (SKOLEM!)
  ((""(EXPAND "Alw")
    ((""(SKOLEM!)
      ((""(EXPAND "LV")
        ((""(EXPAND "IMPLIES")
          ((""(FLATTEN)
            ((""(INSTANTIATE -1 "tt!1")
              ((""(PROPAX) NIL)))))))))))))))
 (|theorem_ii_bis| "" (GRIND) NIL) (|theorem_ii_ter| "" (GRIND) NIL)
 (|theorem_iii| "" (EXPAND "Alw")
  ((""(EXPAND "LV")
    ((""(EXPAND "IMPLIES")
      ((""(SKOLEM!)
        ((""(SKOLEM!)
          ((""(EXPAND "Dist")
            ((""(FLATTEN) ((""(GRIND) NIL)))))))))))))))
 (|theorem_iii_bis| "" (EXPAND "Alw")
```

```
 ((""  (EXPAND "NOW")
   ((""  (EXPAND "Dist")
     ((""  (SKOLEM!)
       ((""  (FLATTEN)
         ((""  (INSTANTIATE -1 "t!1")
           ((""  (GROUND)
             ((""  (EXPAND "IMPLIES")
               ((""  (PROPAX) NIL)))))))))))))))))
(|theorem_iv| "" (TRIO-SIMP) ((""  (SKOSIMP) ((""  (GRIND) NIL)))))
(|theorem_v| "" (GRIND) NIL) (|theorem_v_bis| "" (GRIND) NIL)
(|theorem_vi_b_correct| "" (EXPAND "Futr")
 ((""  (EXPAND "Past")
   ((""  (EXPAND "LV")
     ((""  (SKOLEM!)
       ((""  (EXPAND "NOW")
         ((""  (EXPAND "AND")
           ((""  (EXPAND "IMPLIES")
             ((""  (FLATTEN) ((""  (PROPAX) NIL)))))))))))))))))
(|theorem_vi_c_correct| "" (EXPAND "Futr")
 ((""  (EXPAND "NOW")
   ((""  (EXPAND "Past")
     ((""  (SKOLEM!)
       ((""  (EXPAND "LV")
         ((""  (EXPAND "AND")
           ((""  (EXPAND "IMPLIES")
             ((""  (FLATTEN) ((""  (PROPAX) NIL)))))))))))))))))
(|theorem_vii_a| "" (EXPAND "Dist")
 ((""  (SKOLEM!)
   ((""  (EXPAND "NOW")
     ((""  (EXPAND "IMPLIES")
       ((""  (EXPAND "FA")
         ((""  (FLATTEN) ((""  (PROPAX) NIL)))))))))))))
(|theorem_vii_b| "" (GRIND) NIL)
(|theorem_vii_a_rev| "" (SKOLEM!)
 ((""  (EXPAND "NOW")
   ((""  (EXPAND "Dist")
     ((""  (EXPAND "FA")
       ((""  (SKOLEM!)
         ((""  (EXPAND "IMPLIES")
           ((""  (FLATTEN) ((""  (SKOLEM!) NIL)))))))))))))))
(|theorem_vii_b_rev| "" (SKOLEM!)
 ((""  (FLATTEN)
   ((""  (EXPAND "Dist")
     ((""  (EXPAND "NOW")
       ((""  (EXPAND "EX") ((""  (PROPAX) NIL)))))))))))
(|theorem_viii| "" (SKOLEM!)
 ((""  (FLATTEN)
   ((""  (EXPAND "Dist")
     ((""  (EXPAND "Alw") ((""  (SKOLEM!) ((""  (INST?) NIL)))))))))))
(|inverse_add_time| "" (GRIND) NIL)
```

```
(|theorem_viii_rev| "" (SKOLEM!)
 ((""  (EXPAND "Dist")
   ((""  (EXPAND "Alw")
     ((""  (SKOSIMP)
       ((""  (SKOLEM!)
         ((""  (INSTANTIATE -1 "tt!1 - t!1")
           ((""  (LEMMA "inverse_add_time" ("t1" "t!1"))
             ((""  (REPLACE -2) ((""  (PROPAX) NIL))))))))))))))))
(|induct_mult| "" (GROUND)
 ((""  (LEMMA "NAT_induction") ((""  (GRIND) ((""  (POSTPONE) NIL)))))))
(|theorem_ix| "" (EXPAND "NOW")
 ((""  (EXPAND "LV")
   ((""  (SKOLEM!)
     ((""  (EXPAND "Alw")
       ((""  (EXPAND "Futr")
         ((""  (EXPAND "FA")
           ((""  (EXPAND "IMPLIES")
             ((""  (FLATTEN)
               ((""  (SKOLEM!)
                 ((""  (INDUCT-AND-REWRITE! "tt")
                   ((""  (GROUND)
                     ((""  (FLATTEN)
                       ((""  (LEMMA "NAT_induction")
                         ((""
                           (INST - "(LAMBDA (n : nat): a!1(n * p!1))")
                           ((""  (GROUND)
                             (("1" (GRIND) NIL)
                              ("2"
                               (POSTPONE)
                               NIL))))))))))))))))))))))))))))))))
(|theorem_ix_bis| "" (SKOSIMP)
 ((""  (EXPAND "Alw")
   ((""  (SKOSIMP)
     ((""  (EXPAND "LV")
       ((""  (EXPAND "Futr")
         ((""  (EXPAND "FA")
           ((""  (EXPAND "IMPLIES")
             ((""  (EXPAND "AND")
               ((""  (FLATTEN)
                 ((""  (INDUCT-AND-REWRITE! "tt")
                   ((""  (POSTPONE) NIL))))))))))))))))))))
(|inverse_add_time_const2| "" (GRIND) NIL)
(|theorem_x| "" (EXPAND "NOW")
 ((""  (EXPAND "LV")
   ((""  (EXPAND "Alw")
     ((""  (EXPAND "AlwF_i")
       ((""  (EXPAND "WithinF_ii")
         ((""  (SKOSIMP)
           ((""  (EXPAND "AND")
             ((""  (EXPAND "IMPLIES")
```

```
                ((""  (FLATTEN)
                  (("" (EXPAND "Futr")
                    (("" (SKOSIMP)
                      (("" (INST -2 "0")
                        (("" (BDDSIMP)
                          (("" (INST 1 "-nnt!1 + p!1")
                            (("1" (USE "inverse_add_time_const2")
                              (("1" (REPLACE 1) (("1" (PROPAX) NIL)))))
                             ("2" (GRIND)
                              (("2"
                                (POSTPONE)
                                NIL)))))))))))))))))))))))))))))))))
(|theorem_xi| "" (SKOLEM!)
 (("" (EXPAND "NOW")
   (("" (FLATTEN)
     (("" (EXPAND "Futr")
       (("" (EXPAND "Lasts_ii")
         (("" (EXPAND "WithinF_ii")
           (("" (SKOLEM!)
             (("" (INSTANTIATE 1 "t!1 - it!1")
               (("" (GROUND) NIL)))))))))))))))))
(|theorem_xii| "" (EXPAND "NOW")
 (("" (EXPAND "LV")
   (("" (EXPAND "Alw")
     (("" (EXPAND "AlwF_e")
       (("" (EXPAND "WithinF_ii")
         (("" (SKOSIMP)
           (("" (EXPAND "IMPLIES")
             (("" (EXPAND "Futr")
               (("" (SKOSIMP)
                 (("" (SKOLEM!)
                   (("" (GROUND)
                     (("" (EXPAND "AND")
                       (("" (SKOSIMP)
                         (("" (INST -2 "0")
                           (("" (BDDSIMP)
                             ((""
                               (SKOSIMP)
                               ((""
                                 (INST 1 "it!1 -pt!1 + q!1")
                                 (("1"
                                   (LEMMA "inverse_add_time_const2")
                                   (("1"
                                     (INST -1 "pt!1" "q!1")
                                     (("1"
                                       (REPLACE -1)
                                       (("1"
                                         (REPLACE 1)
                                         (("1" (PROPAX) NIL)))))))))
                                  ("2"
```

```
                                   (GRIND)
                                   (("2"
                                     (POSTPONE)
                                     NIL)))))))))))))))))))))))))))))))))))))
(|theorem_xiii| "" (SKOSIMP)
 (("" (TRIO-SIMP)
   (("" (EXPAND "NOW")
     (("" (EXPAND "LV")
       (("" (EXPAND "Futr")
         (("" (EXPAND "Lasted_ii")
           (("" (EXPAND "OR")
             (("" (EXPAND "AlwF_e")
               (("" (EXPAND "FA")
                 (("" (EXPAND "NOT")
                   (("" (POSTPONE) NIL)))))))))))))))))))))
(|theorem_xiv| "" (AUTO-REWRITE-DEFS :ALWAYS? T)
 (("" (ASSERT)
   (("" (SKOLEM-TYPEPRED)
     (("" (FLATTEN)
       (("" (SKOLEM-TYPEPRED)
         (("" (SKOLEM-TYPEPRED)
           (("" (FLATTEN)
             (("" (ASSERT)
               (("" (INST? :IF-MATCH T :POLARITY? NIL)
                 NIL)))))))))))))))))
(|inverse_add_time_const| "" (GRIND) NIL)
(|theorem_xv| "" (EXPAND "Futr")
 (("" (SKOSIMP)
   (("" (EXPAND "NOW")
     (("" (EXPAND "LV")
       (("" (EXPAND "IMPLIES")
         (("" (EXPAND "FA")
           (("" (FLATTEN)
             (("" (EXPAND "AND")
               (("" (EXPAND "WithinF_ee")
                 (("" (SKOSIMP)
                   (("" (SKOLEM!)
                     (("" (INSTANTIATE 1 "x!1 + it!1")
                       (("1"
                         (LEMMA "inverse_add_time_const"
                          ("t1" "x!1" "t2" "it!1"))
                         (("1" (REPLACE -1) (("1" (PROPAX) NIL)))))
                        ("2" (GRIND) NIL))))))))))))))))))))))))))
```