PUC

Monografias em Ciência da Computação
nº 25/91

On the Theorem Prover Based on Game -Theoretic Semantics

Edward Hermann Haeusler

Departamento de Informática

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO

RUA MARQUÊS DE SÃO VICENTE, 225 - CEP 22453

RIO DE JANEIRO - BRASIL

PUC RIO - DEPARTAMENTO DE INFORMÁTICA

Monografias em Ciência da Computação, Nº 25/91
Editor: Carlos J. P. Lucena Novembro, 1991

On the Theorem Prover Based on Game-Theoretic Semantics*

Edward Herman Haeusler

* This work has been sponsored by the Secretaria de Ciência e Tecnologia da Presidência da República Federativa do Brasil.

Abstract

The Game-Theoretic approach for semantics was created with the purpose of providing meaning to formulae where the traditional semantics have some drawbacks. Hintikka, however, used this approach to provide an alternative semantics w.r.t. the traditional one. The nice feature of this semantics is its quasi-algorithmic specification. This feature has raised some ideas around theorem provers construction.

This report describes a theorem prover based on game-theoretic semantics and shows its soundness. However, as a conclusion of this work some proof-theoretic reasons are pointed out in order to show the inherent incompleteness of this kind of theorem provers when they implement fully the symmetry of the game, that is, the power of proving either the formulae or its negation without submiting each of them (only the formulae is submited).

Key-Words : Mechanical Theorem Proving, Game-Theoretic Semantics.

Resumo

A abordagem de jogos para dar semântica a linguagens lógicas surgiu com o propósito de atribuir-se significado a fórmulas onde a semântica tradicional não é adequada. Hintikka usou esta abordagem como uma via alternativa de atribuição de significado em relação a tradicional. A característica mais interessante desta é seu aspecto quasi-algorítmico. Deste aspecto surgiram ideas em torno da construção de provadores de teoremas.

Este relatório descreve um provador de teoremas baseado na abordagem citada acima e mostra sua corretude com respeito a lógica clássica. Entretanto , com base em considerações de teoria da prova, a incompletude inerente deste tipo de provador, quando a simetria do jogo é mantida, é argumentada. Entendemos por simetria a capacidade de em um único passo provar-se ou a fórmula submetida ou sua negação, resultado este dependendo do vencedor do jogo.

Palavras-Chave: Prova automática de teoremas, Semântica de jogos.

Contents

0.1 Introduction

This report has the purpose of documentation of an experiment done by the author, as a member of the research group of prof. Tarcisio H. Pequeno at PUC/RJ, with the use of game-theoretic semantics for defining a theorem prover for first order classical logic. This experiment culminates with the implementation in LISP of a theorem prover based on game-theoretic semantics improved by some strategies [Rio89]. Further a logic programming environment was implemented using the game-theoretic theorem prover ([Rio89]) as its inference motor.

Here, for the sake of documentation, it is described the theorem prover and shown its soundness by means of a translation process to Natural Deduction. It is also given reasons for its incompleteness based on proof-theoretic arguments. That was the reason for giving up this approach to theorem provers definition. However, some arising questions produced by this experience gave motivation to a methodology, and further, a system for Natural Deduction Theorem Provers contruction [Hae90].

In section I the Theorem Prover is described. In section II its soundness is proved. In section III its incompleteness is commented. However, as this report has only the purpose of documentation, the interested reader should read also the articles ([HP87], [HP88b], [HP88a]) published during the development of the theorem prover, which contain more details concerning the intuition behind it.

0.2 The Definition of the Theorem Prover

0.2.1 The High-game "*Agame*" (Analytical game)

- Base = Knowledge base.
- Relev = Set of relevance of a formula with respect to the Base .
- Relevdin = Dynamic relevance.
- $\vee$-ref = referee of the jogo-$\vee$.
- $\wedge$-ref = referee of the jogo-$\wedge$.
- $\rightarrow$-ref = referee of the jogo-$\rightarrow$.
- atom-ref = referee of the atomic game.

Formally Base is a set of formulas and Relevdin is a subset of it. The referees are sets of pairs of the type (A, J), where A is a formula and J is a player of the game. We name the players as *Nature* and *Myself*. The idea is that *Nature* tries to prove the negation of the formula under game and *Myself* tries to prove the formula itself. It may occurs in the game the case where neither *Nature* nor *Myself* wins, in this case we say that a draw occurred. We name $adv(J)$ the opponent of J, where if J =*Nature* then $adv(J)$ =*Myself* and vice-versa.

Functionally $Agame$ has a pair (J, S) as result, where J is the player who won, and S is a substitution that we had to do in the formula under game such that J would have won. S is a substitution over existential variables.

We will use the projections π_1 and π_2 to obtain the first element of a pair and second element respectively.

1. $Agame(A, J) = Bgame'(A, Relev(A) - RelevDin, J)$.
   with
   $Atom - ref = Atom - ref \cup \{f - ree(B, adv(J))$ such that $B \in DD(A)\}$
   Where Relev(A) is the set of relevance of A
   with respect to the Base.
2. $Agame(\neg A, J) = Agame(A, adv(J))$
3. $Agame(A_1 \vee A_2, J) = (Winner, S)$
   (a) If $\pi_1(Agame(A_1, J)) = adv(J)$ and
   $\pi_1(Agame(\pi_2(Agame(A_1, J))A_2, J)) = adv(J)$
   then
   $Winner = adv(J)$
   $S = \pi_2(Agame(\pi_2(Agame(A_1, J))A_2, J))$.
   (b) If $\pi_1(Agame(A_2, J)) = adv(J)$ and
   $\pi_1(Agame(\pi_2(Agame(A_2, J))A_1, J)) = adv(J)$
   then
   $Winner = adv(J)$
   $S = \pi_2(Agame(\pi_2(Agame(A_2, J))A_1, J))$.
   with $\vee$-ref = $\vee$-ref $\cup$ $\{f - ree(B, adv(J))$ / $B \in DD(A_2)\}$
   (c) If $\pi_1(Agame(A_1, J)) = J$ then
   $Winner = J$
   $S = \pi_2(Agame(A_1, J))$.
   (d) If $\pi_1(Agame(A_2, J)) = J$ then
   $Winner = J$
   $S = \pi_2(Agame(A_2, J))$.
   (e) If nothing above occurs then
   $Winner$ = nobody
   S = undefined

   note : In these cases $\pi_2(Agame(A_1, J))$ must be the most general substitution in the game (among all the alternatives of the following move).

4. $Agame(A_1 \wedge A_2, J) = (Winner, S)$
   (a) If $\pi_1(Agame(A_1, J)) = J$ and
   $\pi_1(Agame(\pi_2(Agame(A_1, J))A_2, J)) = J$
   then
   $Winner = J$
   $S = \pi_2(Agame(\pi_2(Agame(A_1, J))A_2, J))$.
   (b) If $\pi_1(Agame(A_2, J)) = J$ and
   $\pi_1(Agame(\pi_2(Agame(A_2, J))A_1, J)) = J$
   then
   $Winner = J$
   $S = \pi_2(Agame(\pi_2(Agame(A_2, J))A_1, J))$.
   with $\wedge$-ref = $\wedge$-ref $\cup$ $\{f - ree(B, J) \ / \ B \in DD(\neg A_2)\}$
   (c) If $\pi_1(Agame(A_1, J)) = adv(J)$ then
   $Winner = adv(J)$
   $S = \pi_2(Agame(A_1, J))$.
   (d) If $\pi_1(Agame(A_2, J)) = adv(J)$ then
   $Winner = adv(J)$
   $S = \pi_2(Agame(A_2, J))$.
   (e) If nothing above occurs then
   $Winner$ = nobody
   S = undefined

   note : In these cases $\pi_2(Agame(A_1, J))$ must also be the most general substitution in the game (among all the alternatives of the following move).

5. $Agame(\forall x A(x), J) = (Winner, S)$
   (a) If $\pi_1(Agame(A(fx_1, \ldots, x_k), J) = J$ then
   $Winner = J$
   $S = \pi_2(Agame(A(fx_1, \ldots, x_k), J))$
   Where f is a new functional symbol and the x_i's are all of the free variables occuring in $\forall x A(x)$.
   (b) If $\pi_1(Agame(A(e - x), J) = adv(J)$ then
   $Winner = adv(J)$
   $S = \pi_2(Agame(A(e - x), J))$
   (c) If nothing above occurs then
   $Winner$ = nobody
   S = undefined

6. $Agame(\exists x A(x), J) = (Winner, S)$
   (a) If $\pi_1(Agame(A(e-x), J) = J$ then
   $Winner = J$
   $S = \pi_2(Agame(A(e-x), J))$
   (b) If $\pi_1(Agame(A(fx_1, \ldots, x_k), J) = adv(J)$ then
   $Winner = adv(J)$
   $S = \pi_2(Agame(A(fx_1, \ldots, x_k), J))$
   Where f is a new functional symbol and the x_i's are all of the free variables occuring in $\exists x A(x)$.
   (c) If nothing above occurs then
   $Winner$ = nobody
   S = undefined
7. $Agame(A \rightarrow B, J) = (Winner, S)$
   (a) If $\pi_1(Agame(A, J)) = adv(J)$ then
   $Winner = J$
   $S = \pi_2(Agame(A, J))$.
   With $\rightarrow$-ref = $\rightarrow$-ref $\cup$
   (b) If $\pi_1(Agame(A, J)) = J$ and
   $\pi_1(Agame(\pi_2(Agame(A, J))B, J)) = adv(J)$
   then
   $Winner = adv(J)$
   $S = \pi_2(Agame(\pi_2(Agame(A, J))B, J))$.
   With $\rightarrow$-ref = $\rightarrow$-ref $\cup$
   (c) If $\pi_1(Agame(B, J)) = J$ then
   $Winner = J$
   $S = \pi_2(Agame(B, J))$.
   With $\rightarrow$-ref = $\rightarrow$-ref $\cup$ $\{f - ree(C, J) \ / \ C \in DD(\neg A)\}$ and if does not occur in A any existential variable then $Base = \{A\} \cup Base$
   (d) If nothing above occurs then
   $Winner$ = nobody
   S = undefined

0.2.2 The Low-Game "$Bgame$" (Base Game)

Functionally $Bgame$' has an atomic formula and the player who is playing as arguments and a pair (S, J) as result whose first element of it is the winner and the second one is a substitution over the existential variables of the atomic formula which is argument of the game. Besides of the arguments that we

have mentioned there is the set of relevance which represents the subset of the knowledge base where the predicative symbol present in the atomic formula appears.

Dymanic Relevance. We will see later that it is possible the case of a Low-Game to depend on the result of a High-Game. So this High-Game must depend on another Low-Game. Well, the set of relevance of the last Low-Game will have to take into account the fact that one formula of the set of relevance has already been in context which is the formula of the first Low-Game. So we associate to each Bgame a set of formulas that has already been used before the current movement of the game. We call this the set of dynamic relevance (RelevDin). And it will be useful to build the set of relevance when a High-Game calls a Low-Game.

The reader should note that this set of dynamic relevance must be informed from one movement of the game to another. Well, as in the case of the movement to be from one High-Game to another High-Game it is unnecessary to make this information explicit. But, as we have already seen, in the case of a movement from one High-Game to a Low-Game we must do the information explicit (see item i of the definition of the High-Game). In the case of a movement from a Low-Game to a Low-Game the information does not change and no explicity information transmition is necessary. And in the case of a movement from a Low-Game to a High-Game we let implicit the fact that the relevance of any High-Game from now on will be this last dynamic relevance informed by the Low-Game to the High-game. So we will use RelevDin to denote the dynamic relevance before the moving and RelevDin to denote the dynamic relevance after the moving

- $Bgame'(A, \Gamma, J) = (J', S)$, iff there are $\alpha \in \Gamma$ such that $Bgame(A, \alpha, J, T) = (J', S)$.
- Where RelevDin for the above $Bgame$ is RelevDin $\cup \{\alpha\}$.

If this is not the case, then *Bgame'* will have (nobody, undefined) as result.
The functional character of $Bgame$ is such that :

> $Bgame(A, \alpha, J, Flag)$ means that the game is being played with respect to the atomic formula A, the formula α, the player J and the $Flag$ meaning that the formula α is or not under the scope of a negation.

The definition of $Bgame$ is such that :

1. $Bgame(A, B, J, p) = (S, J)$, if and only if S is a substitution over the existential variables of A and over the variables of B (we assume that variables are not existential variables and vice-versa) such that $SA = SB$.
2. $Bgame(A, \neg B, J, p) = Bgame(A, B, adv(J), \neg p)$.
3. $Bgame(A, B_1 \wedge B_2, J, p) =$
$$jogo - \wedge(A, B_1, B_2, J, T), \text{ if } p = T.$$
$$jogo - \vee(A, B_1, B_2, J, F), \text{ if } p = F.$$

4. $Bgame(A, B_1 \vee B_2, J, p) =$
$$jogo - \vee(A, B_1, B_2, J, T), \text{ if } p = T.$$
$$jogo - \wedge(A, B_1, B_2, J, F), \text{ if } p = F.$$
5. $Bgame(A, \forall x B(x), J, p) =$
$$jogo - \forall(A, B(x), J, T), \text{ if } p = T.$$
$$jogo - \exists(A, B(x), J, F), \text{ if } p = F.$$
6. $Bgame(A, \exists x B(x), J, p) =$
$$jogo - \exists(A, B(x), J, T), \text{ if } p = T.$$
$$jogo - \forall(A, B(x), J, F), \text{ if } p = F.$$
7. $Bgame(A, B \rightarrow C, J, p) = jogo - \rightarrow (A, B, C, J, p)$,

Remark : In the following we use $S_{\forall\exists}$ to denote the substitution that replaces all variables by existential variables. By example, x is replaced by $e - x$. We also use two auxiliary functions , namely *player* and Ng, defined as following :

- $Player(J, T) = adv(J)$ and $Player(J, F) = J$
- $Ng(B, T) = B$ and $Ng(B, F) = \neg B$.

Thus, we have the following definitions for $jogo - \vee, jogo - \wedge, jogo - \forall, jogo - \exists$ and $jogo - \rightarrow$:

1. jogo-$\vee(A, B_1, B_2, J, p) = (Winner, S)$
   (a) If $\pi_1(Bgame(A, B_1, J, p)) = J$ and
   $refs - in(S_{\forall\exists}\pi_2(Bgame(A, B_1, J, p))B_2, player(J, p)) = (J, S')$
   then
   $Winner = J$
   $S = S' \circ S_{\forall\exists}\pi_2(Bgame(A, B_1, J, p))$.
   (b) If $\pi_1(Bgame(A, B_1, J, p)) = J$ and
   $Agame(S_{\forall\exists}\pi_2(Bgame(A, B_1, J, p))Ng(\neg B_2, p), J) = (J, S')$
   then
   $Winner = J$
   $S = S' \circ S_{\forall\exists}\pi_2(Bgame(A, B_1, J, p))$
   (c) If $\pi_1(Bgame(A, B_2, J, p)) = J$ and
   $refs - in(S_{\forall\exists}\pi_2(Bgame(A, B_2, J, p))B_1, player(J, p)) = (J, S')$
   then
   $Winner = J$
   $S = S' \circ S_{\forall\exists}\pi_2(Bgame(A, B_2, J, p))$.

(d) If $\pi_1(Bgame(A, B_2, J, p)) = J$ and
$Agame(S_{\forall\exists}\pi_2(Bgame(A, B_2, J, p))Ng(\neg B_1, p), J) = (J, S')$
then
$Winner = J$
$S = S' \circ S_{\forall\exists}\pi_2(Bgame(A, B_2, J, p))$

(e) If $\pi_1(Bgame(A, B_1, J, p)) = adv(J)$ and
$refs - in(\ S_{\forall\exists}\pi_2(Bgame(A, B_1, J, p))B_2,$
$player(adv(J), p)) = (adv(J), S')$
then
$Winner = adv(J)$
$S = S' \circ S_{\forall\exists}\pi_2(Bgame(A, B_1, J, p))$

(f) If $\pi_1(Bgame(A, B_1, J, p)) = adv(J)$ and
$Agame(\ S_{\forall\exists}\pi_2(Bgame(A, B_1, J, p))Ng(\neg B_2, p),$
$adv(J)) = (adv(J), S')$
then
$Winner = adv(J)$
$S = S' \circ S_{\forall\exists}\pi_2(Bgame(A, B_1, J, p))$

(g) If $\pi_1(Bgame(A, B_2, J, p)) = adv(J)$ and
$refs - in(\ S_{\forall\exists}\pi_2(Bgame(A, B_2, J, p))B_1,$
$player(adv(J), p)) = (adv(J), S')$
then
$Winner = adv(J)$
$S = S' \circ S_{\forall\exists}\pi_2(Bgame(A, B_2, J, p)).$

(h) If $\pi_1(Bgame(A, B_2, J, p)) = adv(J)$ and
$Agame(\ S_{\forall\exists}\pi_2(Bgame(A, B_2, J, p))Ng(\neg B_1, p),$
$adv(J)) = (adv(J), S')$
then
$Winner = adv(J)$
$S = S' \circ S_{\forall\exists}\pi_2(Bgame(A, B_2, J, p))$

(i) All the others cases have $(nobody, undefined)$ as value.

2. $jogo - \wedge(A, B_1, B_2, J, p) = (Winner, S)$

- If Some of the games $Bgame(A, B_i, J, p)(i = 1, 2)$
has a winner
then
$(Winner, S) = Bgame(A, B_j, J, p)$, where
this game is the one
which has winner.

3. $jogo - \forall(A, B(x), J, p) = Bgame(A, B(x), J, p)$
4. $jogo - \exists(A, B(x), J, p) = Bgame(A, B(fy_1, \ldots, y_n), J, p)$, where f is a new functional symbol in the game and the y_i's are all of the free variables occuring in $B(x)$ not taking into account x itself.
5. $jogo - \rightarrow (A, B, C, J, p) = (Winner, S)$

If $p = T$ then

Ta If $\pi_1(Bgame(A, C, J, T)) = J$ and
$Agame(\pi_2(Bgame(A, C, J, T))S_{\forall\exists}B, J) = (J, S')$
then
$Winner = J$
$S = S' \circ S_{\forall\exists}\pi_2(Bgame(A, C, J, T))$.

Tb If $\pi_1(Bgame(A, C, J, T)) = adv(J)$ and
$Agame(\pi_2(Bgame(A, C, J, T))S_{\forall\exists}B, adv(J)) = (adv(J), S')$
then
$Winner = adv(J)$
$S = S' \circ S_{\forall\exists}\pi_2(Bgame(A, C, J, T))$

Tc If $\pi_1(Bgame(A, B, adv(J), F)) = adv(J)$ and
$Agame(\pi_2(Bgame(A, B, adv(J), F))S_{\forall\exists}C, adv(J)) = (J, S')$
then
$Winner = adv(J)$
$S = S' \circ S_{\forall\exists}\pi_2(Bgame(A, B, adv(J), F))$

Td If $\pi_1(Bgame(A, B, adv(J), F)) = J$ and
$Agame(\pi_2(Bgame(A, B, adv(J), F))S_{\forall\exists}C, J)) = (adv(J), S')$
then
$Winner = J$
$S = S' \circ S_{\forall\exists}\pi_2(Bgame(A, B, adv(J), F))$

Te If $\pi_1(Bgame(A, C, J, T)) = J$ and
$refs - in(\pi_2(Bgame(A, C, J, T))S_{\forall\exists}B, J) = (J, S')$
then
$Winner = J$
$S = S' \circ S_{\forall\exists}\pi_2(Bgame(A, C, J, T))$.

Tf If $\pi_1(Bgame(A, C, J, T)) = adv(J)$ and
$refs - in(\pi_2(Bgame(A, C, J, T))S_{\forall\exists}B, adv(J)) = (adv(J), S')$
then
$Winner = adv(J)$
$S = S' \circ S_{\forall\exists}\pi_2(Bgame(A, C, J, T))$.

Tg If $\pi_1(Bgame(A, B, adv(J), F)) = adv(J)$ and
$refs - in(\pi_2(Bgame(A, B, adv(J), F))S_{\forall\exists}C, adv(J)) = (J, S')$
then
$Winner = adv(J)$
$S = S' \circ S_{\forall\exists}\pi_2(Bgame(A, B, adv(J), F))$.

Th If $\pi_1(Bgame(A, B, adv(J), F)) = J$ and
$refs - in(\pi_2(Bgame(A, B, adv(J), F))S_{\forall\exists}C, J) = (adv(J), S')$
then
$Winner = J$
$S = S' \circ S_{\forall\exists}\pi_2(Bgame(A, B, adv(J), F))$.

If p = F then

Fa If $Bgame(A, C, J, F)$ has winner
then
$(Winner, S) = Bgame(A, C, J, F)$.

Fb If $Bgame(A, B, J, T)$ has winner
then
$(Winner, S) = Bgame(A, B, J, T)$.

0.2.3 The referees

This theorem prover is based on the one proposed by Jackson [Jac87] (at least in propositional logic). However the game proposed by Jackson has in some cases ended up in loop. The reason is that in these cases we have an inference of a part of the formula, based on another part of it. The typical case is when we need a proof of $\neg A$ to prove B. Well if our goal is to prove $A \vee B$ then with these facts we have already proved it. The function of the referees in the game is to detect these cases giving to the game the right winner. The function $Refs - in$ consults the $\wedge$-ref, the $\vee$-ref, the $\rightarrow$-ref and the *atom*-ref intending to detect the cases mentioned. It receives the formula and the player that will play the movement in the context of the *Agame* and verifies its (of the pair (formula, player)) presence in some of the referees. We use the function $f - ree$ in order to give conditions to the testing below. This function only avoid negative literals in the referee structure, by putting the positive ones and replacing the associated player for his opponent. So :

$$f - ree(A, J) = \begin{cases} (A, J) & \text{if } A \text{ is a positive literal} \\ (B, adv(J)) & \text{if } A = \neg B \end{cases}$$

In the following we use $A \in \Gamma$ to denote the testing whether some instantiation of A (a formula) belongs to Γ (a set of formulae).

$Refs - in(A, J) = (J', S')$
If $((A, J) \in \vee$-ref) or
$((A, J) \in \wedge$-ref) or
$((A, J) \in \rightarrow$-ref) or
$((A, J) \in atom$-ref)
then
$J' = adv(J)$
S' is such that $S'A$ belongs to
some of the referees.

General Remarks :

- The reader should be careful with the definitions because they are indeterministic, i.e., more than one rule may occurs but if the results are different then the knowledge base must be inconsistent.
- Another thing to observe. About the test of membership which the referees do, we should remenber that ,in all referees, ($\vee$-ref, *atom*-ref, etc) we have formulas representing DD's and therefore this test isn't an equality test, but rather an instantiation test (look into the definiton of DD below just in the cases of quantified formulas).

0.2.4 The Deductive Determinant DD

This notion was defined firstly in [Hae87] and is recursively defined as :

- $DD(A) = \{A\}$ if A is a literal.
- $DD(A \rightarrow B) = DD(B)$.
- $DD(A \vee B) = DD(A) \cup DD(B)$.
- $DD(A \wedge B) = DD(A) \star DD(B)$.
- $DD(\neg\neg A) = DD(A)$.
- $DD(\neg(A \vee B)) = DD(\neg A) \star DD(\neg B)$.
- $DD(\neg(A \wedge B)) = DD(\neg A) \cup DD(\neg B)$.
- $DD(\neg(A \rightarrow B)) = DD(A) \star DD(\neg B)$.
- $DD(\forall x A(x)) = \{A(y) \ / \ y$ is a variable of the language $\}$.
- $DD(\exists x A(x)) = \{A(t) \ / \ t$ is a term of the language $\}$.

Where $\{A_1, \ldots, A_k\} \star \{B_1, \ldots, B_m\} = \{A_1 \wedge B_1, A_1 \wedge B_2, \ldots, A_k \wedge B_m\}$.

0.3 The Soundness of the Theorem Prover

In this section we show the relationship between the theorem prover and the system of Natural Deduction (N.D.) due to Prawitz [Pra65]. The relationship is total, i.e., for each theorem proved by the theorem prover we can build up a deduction in N.D. Indeed the building process is a translation one. Thus we show the soundness of the theorem prover by means of that relationship, for the soundness of N.D. is a well-known result. This relationship was used in [HP88a].

We will not stress all the items of the game definition, for there is a lot of analogy in the treatment to be shown here. However, the detailed translation process, from game into N.D., can be found in [Nun88], where is described a formalization of it in the language PROLOG.

As it can be noted, the game works out by looking at the main connective of the formula, for each move is determined by that connective. Natural Deduction works out likewise, the only difference being that in it we work towards the formula we want to show, while in games we work from it. Smullyan [Smu68] showed this is only a viewpoint matter when he relates tableaux to Sequent Calculus proofs by seeing the former bottom-up. Here we use a version of this method in order to show a translation from games into Natural Deduction.

In a rule of the Agame there can be seen two kinds of items : those that define as winner the same player that began the game and those that define as winner its opponent. We call them direct and indirect items respectively.

In a rule of the Bgame we also have two different kinds of items : those which have the negation flag (last parameter of Bgame) true and those which have it false, informing that there is no negation over the basis formula (second parameter of Bgame) and there is a negation over it respectively. We call the first ones direct items, while the other are called indirect items.

Note that the direct items of an Agame are different in essence of the respective ones of the Bgame. We can associate each direct item of each Agame case to the N.D.'s introduction rule over the respective connective. So, for example :

$$Agame(A \wedge B, J) = (J, S) \text{ iff } Agame(A, J) = (J, S_1) \text{ and } Agame(S_1 B, J) = (J, S)$$

is associated with ,

$$\frac{SA \quad SB}{S(A \wedge B)}$$

,and

$$Agame(A \vee B, J) = (J, S) \text{ iff } Agame(A, J) = (J, S) \text{ or } Agame(B, J) = (J, S)$$

is associated with,

$$\frac{SA}{SA \vee SB} \qquad \frac{SB}{SA \vee SB}$$

, respectively.

In a similar manner we can associate the Bgame's direct rules to N.D. elimination rules. So, for example

$$Bgame(A, B \wedge C, J, T) = (J, S) \text{ iff } Bgame(A, B, J, T) = (J, S) \text{ or } Bgame(A, C, J, T) = (J, S).$$

is associated with the $\wedge$-elimination rule. However, this association is not as straightforward as the first one. If it were the case of $Bgame(A, A \wedge B, J, T) = (J, \emptyset)$ where $\emptyset$ is the empty substitution then the association would be straightforward. But, looking from a global viewpoint, we note that this association is not directly related to the atomic formula in the *Bgame* context (A), rather to a lower degree formula which is associated to another one and so on until a direct association to A is found as an elimination rule of Natural Deduction.

So, by associating the new constants/functions of the Bgame over the $\exists$, as well as the ones passed by the Agame over the $\forall$, to parameters in Natural Deduction proofs, we complete the association for direct rules. So, for example :

$$Agame(\forall x A(x), J) = (J, S) \text{ iff } Agame(A(c), J) = (J, S)$$

is associated to the $\forall$-introduction rule, since c ,being a new constant, cannot be replaced by any term (remind that it is not affected by S), i.e., it can be viewed as a formal parameter which does not occur in any formula that participates in $Agame(A(c), J)$. Moreover, note that if there are free variables occuring in $\forall x A(x)$ then a functional symbol, with arity equals to the numbers of those free variables and depending on then, is need in order to ensure that the parameter which will be replaced by it does not occur in any assumption after the translation. Remember that the free variables occuring in the term formed by the new functional symbol will be replaced during the game sequel. This treatment of paramenters related to skolemization is found in [Sch84].

In the same way :

$$Bgame(A, \exists x B(x, y_1, \ldots, y_n), J, T) = (J, S) \text{ iff } Bgame(A, B(fy_1, \ldots, y_n), J, T) = (J, S)$$

is associated with the $\exists$-elimination rule, for the new constant (functional) symbol c (f) can be viewed as its proper parameter. In the case of the functional symbol, we take the term $Sfy_1, \ldots, y_n$ as playing the role of the proper parameter. There is a problem with that association when the new constant c ,or the whole term $fy_1, \ldots, y_n$, replaces an existencial variable of the *Agame*, hence it may occur in the minor premiss of the $\exists$-elimination. The solution to this problem is discussed in section 2.1.

So the association shown above is such that each direct rule of Agame is associated with the N.D. introduction rule over the same connective, and the direct rules of Bgame are associated with the elimination rules.

Now we see what happens when the referees are included in the game. These cases are viewed as a case of application of a classical axiom and a $\vee$-elimination rule and a deduction built up by using the deductive determinant property [Hae87]. The final result of the association to N.D. in this case is as follows :

$Agame(A \vee B, J) = (J, S)$, for $Agame(A, J) = (J, S)$ and the game used a consultation to the $\vee$-referee about δ, where δ is a deductive determinant of B. So, the translation is :

$$\dfrac{\neg\delta \vee \delta \qquad \dfrac{\begin{array}{c}[\neg\delta]\\ |\\ SA\end{array}}{S(A \vee B)} \qquad \dfrac{\begin{array}{c}[\delta]\\ |\\ SB\end{array}}{S(A \vee B)}}{S(A \vee B)}$$

Where the deduction of SA from δ is the result of the translation into natural deduction of $Agame(A, J) = (J, S)$, and the deduction of SB from δ is obtained by means of a procedure that builds a deduction of a formula from any of its deductive determinants which can be found in [Hae87]. The other referees have similar treatment. It is interesting to point out that even the case of the direct rule over the $\rightarrow$ in the Agame context has as associated natural deduction the $\rightarrow$-introduction rule, for we add to the Base the antecedent of the implication, so we can discharge its (possible) use.

With regard to the negation, we do not associate it to Natural Deduction rules directly. The reason of this is that the game definition does not implement the negation directly, rather by defining how a negated conjunction, a negated disjunction and so on can be deduced. The same happens w.r.t. the Bgame indirect rules, which defines how to deduce from each part of a negated conjuntion, a negated disjunction and so on. Therefore the indirect rules are associated to a derived rule to the negation of each connective, and to the derived rules that say how to eliminate a connective which is within the scope of a negation. For example :

$$\dfrac{\neg SA \quad \neg SB}{\neg S(A \vee B)}$$

and

$$\dfrac{\neg S(A \vee B)}{\neg SA}, \dfrac{\neg S(A \vee B)}{\neg SB}$$

are respectively the N.D. (modified) rules associated with :

- $Agame(A \vee B, J) = (opn(J), S)$ when $Agame(A, J) = (opn(J), S_1)$ and $Agame(S_1 B, J) = (opn(J), S)$, and ;
- $Bgame(A, B \vee C, J, F) = (J', S)$ when either $Agame(A, B, J, F) = (J', S)$ or $Agame(A, C, J, F) = (J', S)$ (J' is any player).

In the sequel we present the whole translation process and prove its correctness.

0.3.1 The Translation

In order to attain the translation into N.D. from a game G we join in a whole deduction the N.D. represention of each rule used by G, according the sequel given by G itself. Thus, making this connection recursively we note that the basic step is just the rule $Bgame(A, A', J, f) = (J, S)$ when the translation is SA (A and A' atomic). We Split the translation into two partial translations, one responsible for $Bgame$ translations and another for $Agame$ translations. We call them $TradA$ and $TradB$ respectively. Since the game is defined by a simultaneous recursion between $Agame$ and $Bgame$, the total translation must also be.

So we have the following desired results about the translation, that prove its correctness :

Lemma 0.3.1 . *Let $Bgame(A, B, J, f) = (J', S)$ be a game that does not call any instance of Agame. Then :*

- *If $J' = J$ and $f = T$ then $TradB(Bgame(A, B, J, f))$ is a deduction of SA from SB (only).*
- *If $J' = J$ and $f = F$ then $TradB(Bgame(A, B, J, f))$ is a deduction of SA from $\neg SB$ (only).*
- *If $J' = opn(J)$ and $f = T$ then $TradB(Bgame(A, B, J, f))$ is a deduction of $\neg SA$ from SB.*
- *If $J' = opn(J)$ and $f = F$ then $TradB(Bgame(A, B, J, f))$ is a deduction of $\neg SA$ from $\neg SB$.*

The proof proceeds by a simple induction over the degree of B.

Proposition 0.3.1 *Let α be a formula and $Base = \Gamma$ a set of formulas. Assume that the utterance of lemma 2.1 holds for all Bgame calls in $Agame(\alpha, J)$. Then :*

- *If $Agame(\alpha, J) = (J, S)$ then $TradA(Agame(\alpha, J))$ is a deduction of $S\alpha$ from Γ.*
- *If $Agame(\alpha, J) = (opn(J), S)$ then $TradA(Agame(\alpha, J))$ is a deduction of $\neg\alpha$ from Γ.*

Proof. By induction on the degree of α. The assumption done in the hypothesis of the proposition guarantees the basic step. The inductive step for each case of the game is done by using the corresponding rule or derived rule (to the negation) in order to construct the desired deduction from the previously contructed , by the induction process, subderivations (deductions of the premisses).

Now we use proposition 2.1 and lemma 2.1 in order to prove the desired result used as a hypothesis in the proposition itself (in terms of $Agame$ calls instead of $Bgame$ calls).

Theorem 0.3.1 *Let $Base = \Gamma$ be a set of formulas and A be an atomic formula. Let $Agame(A, J) = Bgame(A, B, J, T)$ for some $B \in \Gamma$ be a game which has (J', S) as its result. Then :*

- *If $J' = J$ then $TradA(Agame(A, J)) = TradB(A, B, J, T)$ is a deduction of SA from Γ.*
- *If $J' = opn(J)$ then $TradA(Agame(A, J)) = TradB(A, B, J, T)$ is a deduction of $\neg SA$ from Γ.*

Proof. By induction on the number of calls to *Agame* from *Bgame*. The basic step is lemma 2.1. If there is a call to one instance of a Agame, then : If it is not an atomic instance (instance over an atomic formula), it surely calls atomic instances. Well , these instances have a lower value of induction, hence their translations by $TradA$ are of the required type,i.e, we can apply the induction hypothesis. Thus, we use proposition I and get the desired conclusion. It is clear that may happen a consultation to a referee, so in this case we use the schema shown in the first section.

Now by using proposition 2.1 again and theorem 2.2 we get the desired result :

Theorem 0.3.2 *Let α be a formula and $Base = \Gamma$ a set of formulas, then :*

- *If $J' = J$ then $TradA(Agame(\alpha, J))$ is a deduction of $S\alpha$ from Γ.*
- *If $J' = opn(J)$ then $TradA(Agame(\alpha, J))$ is a deduction of $\neg\alpha$ from Γ.*

0.3.2 The permutations on the $\exists$-elimination

The process of translation done by the pair of functions ($TradA$ and $TradB$) are straightforward, since they perform a simple mutual recursion and they build the corresponding tree derivation by using the rules and derived rules to the negation. The only care we have to take is to permute the existential elimination until it obeys the restriction over its proper parameters, what must happen when we get the existential introduction that eliminates the proper parameter. This is the only case when we cannot use the simple way of construction, that is, to connect translated premisses by using a rule and getting the translated conclusion. Moreover we should argue that these permutations are always possible and result in correct N.D. derivations.

The permutation are of the following form for rules with one premiss (the general case has a similar treatment):

$$\dfrac{\dfrac{\dfrac{\begin{matrix}\Pi_1 \\ \exists y Py\end{matrix} \qquad \begin{matrix}[Pb] \\ \Pi_1 \\ C\end{matrix}}{C}}{\begin{matrix}D \\ \Pi_3 \\ \alpha\end{matrix}}}{} \quad \triangleright \quad \dfrac{\begin{matrix}\Pi_1 \\ \exists y Py\end{matrix} \qquad \dfrac{\begin{matrix}[Pb] \\ \Pi_2 \\ C\end{matrix}}{D}}{\begin{matrix}D \\ \Pi_3 \\ \alpha\end{matrix}}$$

where b, or the functional symbol introduced by the corresponding rule of *Bgame* that occurs in b, still occurs in D.

It is clear that there is some place in Π_3 where a $\exists$ is introduced and the permutations are not need any more. After all the permutations are already done we replace the b by a proper (new) parameter of the $\exists$-elimination rule. However, we must ensure that all the $\forall$ introductions were well done, i.e., they

obey the restriction on the $\forall$-introduction. In other words we must assure that in the following result of a permutation:

$$\begin{array}{c}\dfrac{\begin{array}{c}\Pi_1\\ \exists y Py\end{array} \qquad \begin{array}{c}[Pb]\\ \Pi_2\\ \dfrac{Aa}{\forall x Ax}\end{array}}{\forall x Ax}\\ \Pi_3\\ \alpha\end{array}$$

a does not occur in Pb. But we have the following facts :

- In order to a occur in Pb, it (a) must be the result of an instantiation from an $\forall$-Elimination rule in Π_2. Thus, $\exists y Py$ has free variables.
- But, in that case, because of the theorem prover evaluation, b is a term built with a new functional symbol and b depends on the variable eliminated by the mentioned $\forall$-Elimination (call it z).
- Aa still contains b, for otherwise this permutation would not be need any more.
- Thus there is an $\exists$-Introduction over a variable w with regard to b below the showed $\forall$-Introduction. Thus there is a free variable (call it w) occuring in $\forall x Ax$.
- Thus, because of the theorem prover evaluation, a is a functional term built up with a new functional symbol depending on w (possibly among other variables).
- In Π_1 there is an atomic formula which contains a and b, for both occur in Aa and Pb. Call that atomic formula α.
- Let $\alpha = Qt_1, \ldots, t_n$. Because of the theorem prover evaluation, α is the result of the unification of an atomic formula $Qt'_1, \ldots, t'_n$ and $Qt''_1, \ldots, t''_n$. Where for some i and j, t'_i contains $b(\ldots, z, \ldots)$ and t''_j contains $a(\ldots, w, \ldots)$. Moreover, z is replaced by $a(\ldots, w, \ldots)$ and w is replaced by $b(\ldots, z, \ldots)$. However this is impossible.

Thus, a does not occur in Pb and we can replace all the terms with new functional symbols by proper parameters, in order to obtain a correct N.D. proof.

0.3.3 The Soundeness

An immediate consequence of Theorem 2.2 is :

Corollary 0.3.1 . *The game procedure is sound.*

0.4 On the incompleteness

It has already said that the theorem prover is incomplete. Indeed, its incompleteness might be seen as inherent to the proof method, namely the game.

As the reader must have noted, the theorem prover carries out its procedure by analysing the formula into its atomic components and then analysing also the formulae of the knowledge base into its atomic components. After that an unification algorithm is used to check the relation between pairs of atomic components (one obtained from the formula (goal) analysis and the other obtained from the knowledge-base formula analysis). From the process of translation one can note that the theorem prover (or proof method) search for proofs which have an elimination part (E-part) (possibly empty) immediately followed by an introdcution part (I-part). The I-part has only introduction rules and terminates with either the formula wanted to be proved or some minor premiss of the $\rightarrow$-Elimination. The E-part begins from some formula of the knowledge-base. The reader familiar with proof-theory must have noted that the theorem prover is nothing but a searcher for normal proofs.

Based on the discussed in the above paragraph it can clearly be seen the cause of the incompleteness of the method. That is :

1. The method searchs for normal proofs.
2. It replaces the classical absurdity rule by the rule of double negation ($\neg\neg A \vdash A$).
3. There is no Normal Form Theorem for the system with this replacement. ([Pra65]).

Thus, it may be the case to propose to the theorem prover a valid formula which does not have normal proof, obtaining a wrong answer (Failure or LOOP !).

One may also have noted that the use of the double negation plays an essencial role in the game definition. In order to be called a game, the rules must be simetric, i.e., the players can swap places and goals and this change will only affect the name of the winner. Thus, the game-theoretic semantics approach to theorem provers construction does not seem to be as good as it seemed at first sight.

Well, one may think that the problem is with the classical logic. But if a simpler logic as the minimal logic ([Pra65]) is focused a new problem arises , since the game is based on the simetrical role played by the opponents. There is no possibility of elimination of the $\forall$ within the scope of an odd number of negations (the particular case of only one negation gives enough problems). Thus, any formula of the above kind should be regarded as a non-analisable on. The only thing to do is to check for unification with other formula of the same kind, as it is done with atomic formulas. But, this solution does not seem to agreed with the basic idea of game-theoretic semantics, according to which the mentioned kind of formulas should be considered as having a primitive semantics.

Thus, based on the reported experience one had to conclude that the nice features of the game-theoretic semantics, mainly its way to define the semantics of the negation not merely as a metalevel negation, are so atractive to theorem provers construction as non-computational. This should not be surprising, since the

game-theoretic semantics for atomic formulas is (originately) given within the framework of structures for the first-order languages and supposes a model (possibly infinite or even non-denumerable).

0.5 Acknowledgements

The author would like to thank : Tarcisio Pequeno for the encouragement and discussions, Riverson Rios for implementing the prover and for appointing some mistakes during the development and Graca Nunes for implementing the translation process and also for appointing some mistakes in the first version. The author is also thankful to them for the friendship.

Bibliography

[Hae87] Edward H. Haeusler. Automatic theorem proving : An attempt to improve the readability of proofs generated by resolution. In *Methods and Applications of Mathematical Logic.* American Mathematical Society, 1987.

[Hae90] E.H. Haeusler. *Prova automatica de teoremas em dedução natural : Uma abordagem abstrata.* Tese de Doutorado, Depto Inf. PUC/RJ, 1990.

[HP87] Edward H. Haeusler and Tarcisio H. Pequeno. Um provador de teoremas baseado em jogos semanticos. In *Anais do IV Simposio Brasileiro de Inteligencia Artificial*, 1987.

[HP88a] Edward H. Haeusler and Tarcisio H. Pequeno. Explanation by game. In *Proceedings of the I International Symposium on Intelligent Tutoring Systems*, Montreal, Canada, Jun 1988. ACM.

[HP88b] Edward H. Haeusler and Tarcisio H. Pequeno. A theorem prover based on game-theoretic semantics. In *Anais da I Conferencia Internacional Sobre Informatica*, La Habana Cuba, 1988.

[Jac87] Peter Jackson. Implementing a game-theoretical interpretation of logic. Technical report, Dept. of Artificial Intelligence Edinburgh, 1987.

[Nun88] Maria G. Nunes. A bla bla bla in prolog. A Work implemented and tested, 1988.

[Pra65] Dag Prawitz. *Natural Deduction.* Stockholm, 1965.

[Rio89] Jose Riverson A. C. Rios. Reasoning with semantic games. Master's thesis, Dept of Computing of PUC/RJ, 1989. In Portuguese.

[Sch84] D. Schmidt. A programming notation for tactical reasoning. In *Seventh Conference on Automated Reasoning, LNCS 170.* Springer-Verlag, 1984.

[Smu68] Raymond Smullyan. *First Order Logic.* Springer-Verlag, 1968.