SID – Wykład 6
Wnioskowanie w logice I rzędu

Dominik Ślęzak

Wydział Matematyki, Informatyki i Mechaniki UW
slezak@mimuw.edu.pl

Rachunek zdań: zalety i wady

- Rachunek zdań jest deklaratywny: elementy syntaktyki odpowiadają faktom
- Rach. zdań dopuszcza częściową/alternatywną/zanegowaną informację (w przeciwieństwie do większości struktur danych i baz danych)
- Rachunek zdań jest składnikowy: znaczenie $B_{1,1} \wedge P_{1,2}$ wynika ze znaczenia $B_{1,1}$ i $P_{1,2}$
- Znaczenie w rachunku zdań jest niezależne od kontekstu (w przeciwieństwie do języka naturalnego)
- Rachunek zdań ma bardzo ograniczoną moc wyrażania (w przeciwieństwie do języka naturalnego), np. nie da się wyrazić zdania "pułapki powodują wiatr w sąsiednich polach" inaczej niż przez napisanie oddzielnego zdania dla każdego pola

Logika I rzędu

Rachunek zdań zakłada, że świat zawiera tylko fakty, natomiast logika I rzędu (bliższa językowi naturalnemu) zakłada, że świat zawiera:

- Obiekty: ludzie, domy, liczby, teorie, kolory, baseball, wojny, wieki …
- Relacje: czerwony, okrągły, pierwszy, większy niż, w środku, część, ma kolor, posiada, …
- Funkcje: trzeci właściciel, o jeden więcej niż, początek …

Syntaktyka i semantyka

Stałe: *KingJohn*, $0, 1, 2, \ldots$

Predykaty: *Brother*, $>, \ldots$

Funkcje: *Sqrt*, $\ldots$

Zmienne: x, y, a, $b, \ldots$

Łączniki: $\wedge \ \vee \ \neg \ \implies \ \iff$

Równość: $=$

Kwantyfikatory: $\forall \ \exists$

$$\begin{aligned} \text{Zdania atomowe} &= predicate(term_1, \ldots, term_n) \\ &\quad \text{lub } term_i = term_j \\ Term &= function(term_1, \ldots, term_n) \\ &\quad constantorvariable \end{aligned}$$

Zdania złożone są budowane ze zdań atomowych przy pomocy łączników:

$$\neg S, S_1 \wedge S_2, S_1 \vee S_2, S_1 \implies S_2, S_1 \iff S_2$$

Semantyka

Zdania są prawdziwe względem modelu i interpretacji.

Model zawiera ≥ 1 obiekt (element dziedziny) i relacje między nimi.

Interpretacja specyfikuje przyporządkowania:

symbole stałe $\rightarrow$ obiekty

predykaty $\rightarrow$ relacje

symbole funkcyjne $\rightarrow$ relacje funkcyjne

Zdanie atomowe $\underline{predicate(term_1, \ldots, term_n)}$ jest prawdziwe $\iff$ obiekty przyporządkowane do $term_1, \ldots, term_n$ są w relacji przyporządkowanej do *predicate*.

Modele logiki I rzędu: bardzo dużo!

Można wyliczać modele dla danego słownika bazy wiedzy *KB*:

Dla każdej liczby elementów dziedziny n od 1 do ∞
  Dla każdego k-arnego predykatu P_k w słowniku
    Dla każdej możliwej relacji k-arnej na n obiektach
      Dla każdego symbolu stałego C w słowniku
        Dla każdego przyporządkowania C do jednego z n obiektów ...

Określanie logicznych konsekwencji przez wyliczanie modeli jest niepraktyczne!

Kwantyfikatory

Kwantyfikator uniwersalny: $\forall\langle variables\rangle\langle sentence\rangle$

Każdy w Berkeley jest sprytny: $\forall x\ At(x, Berkeley) \implies Smart(x)$

$\forall x\ P$ jest prawdziwe w modelu $m \iff P$ jest prawdziwe z x przyporządkowanym do każdego możliwego obiektu w modelu

---

Kwantyfikator egzystencjalny: $\exists\langle variables\rangle\langle sentence\rangle$

Ktoś w Stanford jest sprytny: $\exists x\ At(x, Stanford) \land Smart(x)$

$\exists x\ P$ jest prawdziwe w modelu $m \iff P$ jest prawdziwe z x przyporządkowanym do pewnego możliwego obiektu w modelu

Tłumaczenie języka naturalnego do I rzędu

Bracia są rodzeństwem

$$\forall x, y\ Brother(x, y) \implies Sibling(x, y).$$

"Rodzeństwo" jest symetryczne

$$\forall x, y\ Sibling(x, y) \iff Sibling(y, x).$$

Czyjaś matka jest kobietą i jego rodzicem

$$\forall x, y\ Mother(x, y) \iff (Female(x) \wedge Parent(x, y)).$$

Kuzyn jest dzieckiem rodzeństwa rodzica

$$\forall x, y\ FirstCousin(x, y) \iff \exists p, ps\ Parent(p, x) \wedge Sibling(ps, p) \wedge Parent(ps, y)$$

Równość

$term_1 = term_2$ jest prawdziwe w danej interpretacji $\iff$ jeśli $term_1$ i $term_2$ przyporządkowane są do tego samego obiektu

Np.

$1 = 2$ i $\forall x \times(Sqrt(x), Sqrt(x)) = x$ są spełnialne

$2 = 2$ jest tautologią

Np. definicja (pełnego) *Sibling* w terminach *Parent*:

$$\forall x, y\ Sibling(x, y) \iff [\neg(x = y \wedge \exists m, f\ \neg(m = f) \wedge Parent(m, x) \wedge Parent(f, x) \wedge Parent(m, y) \wedge Parent(f, y)]$$

Baza wiedzy w języku I rzędu

Agent w świecie Wumpusa odczuwający zapach i wiatr, ale nie obserwujący błysku w chwili $t = 5$:

$Tell(KB, Percept([Smell, Breeze, None], 5))$
$Ask(KB, \exists a\ Action(a, 5))$

tzn. czy z KB wynikają jakieś konkretne akcje w chwili $t = 5$?

Odpowiedź: Tak, $\{a/Shoot\}$ $\leftarrow$ <u>podstawienie</u> (lista powiązań)

Dla danego zdania S i podstawienia σ,
$S\sigma$ oznacza wynik zastosowania σ do S; np.
$S = Smarter(x, y)$
$\sigma = \{x/Hillary, y/Bill\}$
$S\sigma = Smarter(Hillary, Bill)$

$Ask(KB, S)$ zwraca pewne/wszystkie σ takie, że $KB \models S\sigma$

Baza wiedzy: przykład

"Obserwacja"

$$\forall b, g, t \; Percept([Smell, b, g], t) \implies Smelt(t)$$
$$\forall s, b, t \; Percept([s, b, Glitter], t) \implies AtGold(t)$$

Reakcja:

$$\forall t \; AtGold(t) \implies Action(Grab, t)$$

Reakcja z wewnętrznym stanem: czy nie mamy już złota?

$$\forall t \; AtGold(t) \land \neg Holding(Gold, t) \implies Action(Grab, t)$$

Baza wiedzy: przykład

Własności miejsca:

$$\forall x, t \; At(Agent, x, t) \wedge Smelt(t) \implies Smelly(x)$$
$$\forall x, t \; At(Agent, x, t) \wedge Breeze(t) \implies Breezy(x)$$

Blisko pułapek jest wiatr:

Reguła diagnostyczna — wnioskuje przyczynę z efektu

$$\forall y \; Breezy(y) \implies \exists x \; Pit(x) \wedge Adjacent(x, y)$$

Reguła przyczynowa — wnioskuje efekt z przyczyny

$$\forall x, y \; Pit(x) \wedge Adjacent(x, y) \implies Breezy(y)$$

Definicja:

$$\forall y \; Breezy(y) \iff [\exists x \; Pit(x) \wedge Adjacent(x, y)]$$

OBSERWACJA: logika I rzędu ma dużo większą moc wyrażania niż rachunek zdań

Wnioskowanie w logice: historia

450 BC	Stoicy	rachunek zdań, wnioskowanie (prawdopodobnie)
322 BC	Arystoteles	“sylogizmy” (reguły wnioskowanie), kwantyfikatory
1565	Cardano	teoria prawd. (rachunek zdań + niepewność)
1847	Boole	rachunek zdań (ponownie)
1879	Frege	logika I rzędu
1922	Wittgenstein	dowodzenie przez tabele prawdziwości
1930	Gödel/Herbrand	pełne systemy wnioskowania dla logiki I rzędu
1931	Gödel	$\neg\exists$ pełny system wnioskowania dla arytmetyki
1960	Davis/Putnam	praktyczny algorytm dla rachunku zdań
1965	Robinson	praktyczny “algorytm” dla logiki I rzędu – rezolucja

Podstawienia

Podstawienie: Zbiór postaci

$$\{x_1/t_1, \dots x_n/t_n\}$$

gdzie $x_1, \dots, x_n$ są różnymi zmiennymi, natomiast $t_1, \dots, t_n$ są termami.

Dopuszczamy podstawienie puste, oznaczane ε.

Niech θ będzie podstawieniem postaci (1) i niech α będzie wyrażeniem (tzn. formułą lub termem). Piszemy $\alpha\theta$ na oznaczenie wyrażenia powstałego z α w wyniku jednoczesnego zastąpienia wszystkich wolnych wystąpień zmiennych $x_1, \dots, x_n$ termami $t_1, \dots, t_n$.

Na przykład, jeśli $\theta = \{x/Sam, y/Pam\}$, to

$$(Likes(x, y))\theta = Likes(Sam, Pam).$$

Instancjacja uniwersalna

Każda instancjacja uniwersalnie kwantyfikowanego zdania jest logiczną konsekwencją reguły:

$$\frac{\forall v\ \alpha}{Subst(\{v/g\}, \alpha)}$$

dla dowolnej zmiennej v i termu ustalonego g

Instancjacja uniwersalna może być stosowana
kilkakrotnie, żeby dodać nowe zdania;
nowa KB jest logicznie równoważna poprzedniej

Redukcja do rachunku zdań

Załóżmy, że mamy daną następującą bazę wiedzy KB:

$$\forall x\ King(x) \wedge Greedy(x) \implies Evil(x)$$
$$King(John)$$
$$Greedy(John)$$
$$Brother(Richard, John)$$

Instancjując zdania z kwantyfikatorami uniwersalnymi na wszystkie możliwe sposoby otrzymujemy

$$King(John) \wedge Greedy(John) \implies Evil(John)$$
$$King(Richard) \wedge Greedy(Richard) \implies Evil(Richard)$$
$$King(John)$$
$$Greedy(John)$$
$$Brother(Richard, John)$$

Nowa KB jest sprowadzona do języka zdań: symbole zdaniowe to

$$King(John),\ Greedy(John),\ Evil(John),\ King(Richard),\ \text{itd.}$$

Redukcja do rachunku zdań

Fakt: zdanie ustalone jest logiczną konsekwencją nowej KB

$\Longleftrightarrow$

jest logiczną konsekwencją oryginalnej KB

Fakt: każda baza wiedzy bez kwantyfikatorów egzystencjalnych i symboli funkcyjnych może być sprowadzona do języka rachunku zdań, tak aby zachować logiczne konsekwencje

Pomysł:

sprowadź KB i zapytanie do języka rachunku zdań

zastosuj rezolucję dla rachunku zdań

zwróć wynik

Redukcja do rachunku zdań: efektywność

Redukcja do rachunku zdań generuje wiele nieistotnych zdań.
Np.

$\forall x\ King(x) \wedge Greedy(x) \implies Evil(x)$
$King(John)$
$\forall y\ Greedy(y)$
$Brother(Richard, John)$

Fakt $Evil(John)$ wydaje się oczywisty, ale redukcja produkuje wiele faktów takich jak $Greedy(Richard)$, które są nieistotne.

Dla p k-arnych predykatów i n stałych, będzie $p \cdot n^k$ instancjacji!

Unifikacja

Wniosek $Evil(John)$ z poprzedniego slajdu można otrzymać bardziej bezpośrednio, jeśli znajdziemy podstawienie θ takie, że $King(x)$ oraz $Greedy(x)$ pasują do $King(John)$ oraz $Greedy(y)$.

$\theta = \{x/John, y/John\}$ spełnia te wymagania.

$UNIFY(\alpha, \beta) = \theta$ jeśli $\alpha\theta = \beta\theta$

p	q	θ
$Knows(John, x)$	$Knows(John, Jane)$	$\{x/Jane\}$
$Knows(John, x)$	$Knows(y, OJ)$	$\{x/OJ, y/John\}$
$Knows(John, x)$	$Knows(y, Mother(y))$	$\{y/John, x/Mother(John)\}$
$Knows(John, x)$	$Knows(x, OJ)$	*fail*

Unifikacja

Podstawienie θ jest unifikatorem wyrażeń $E_1, \ldots, E_n$ jeśli $E_1\theta = E_2\theta = \ldots = E_n\theta$.

Podstawienie θ jest najogólniejszym unifikatorem wyrażeń $E_1, \ldots, E_n$ jeśli dla każdego unifikatora tych wyrażeń, γ, istnieje podstawienie λ takie, że $\gamma = \theta\lambda$.

Warto szukać jak najogólniejszych unifikacji.
Należy uważać przy przemianowywaniu zmiennych:
Wszystkie kwantyfikowane zmienne muszą być różne.

Uogólniony Modus Ponens

$$\frac{p_1{}',\ p_2{}',\ \ldots,\ p_n{}',\ (p_1 \wedge p_2 \wedge \ldots \wedge p_n \implies q)}{q\theta}$$

gdzie $p_i{}'\theta = p_i\theta$ dla wszystkich i

$p_1{}'$ jest $King(John)$ | p_1 jest $King(x)$

$p_2{}'$ jest $Greedy(y)$ | p_2 jest $Greedy(x)$

θ jest $\{x/John, y/John\}$ | q jest $Evil(x)$

$q\theta$ jest $Evil(John)$

Uogólnione Modus Ponens używa baz wiedzy klauzul definiujących (dokładnie jeden literał pozytywny)

Zakłada się, że wszystkie zmienne są kwantyfikowane uniwersalnie

Przykład

Prawo amerykańskie określa, że sprzedaż broni obcemu narodowi przez Amerykanina jest przestępstwem. Państwo Nono, które jest wrogie, ma pewne pociski. Wszystkie pociski zostały mu sprzedane przez pułkownika Westa, który jest Amerykaninem.

Pokazać, że pułkownik West jest przestępcą.

Baza wiedzy: przykład

...sprzedaż broni obcemu narodowi przez Amerykanina jest przestępstwem:

$$American(x) \wedge Weapon(y) \wedge Sells(x, y, z) \wedge Hostile(z) \implies Criminal(x)$$

Nono ...ma pewne pociski, tzn. $\exists x\ Owns(Nono, x) \wedge Missile(x)$:

$Owns(Nono, M_1)$ i $Missile(M_1)$

...wszystkie pociski zostały mu sprzedane przez pułkownika Westa:

$$\forall x\ Missile(x) \wedge Owns(Nono, x) \implies Sells(West, x, Nono)$$

Pociski są bronią:

$$Missile(x) \implies Weapon(x)$$

Nieprzyjaciel Ameryki uznawany jest za "wrogi":

$$Enemy(x, America) \implies Hostile(x)$$

West, który jest Amerykaninem ...

$American(West)$

Państwo Nono, nieprzyjaciel Ameryki ...

$Enemy(Nono, America)$

Rezolucja

Metodę rezolucji stosuje się dla formuł w postaci klauzulowej.

Klauzula – formuła postaci

$$l_1 \vee \ldots \vee l_n \qquad (n \geq 0)$$

gdzie $l_1, \ldots, l_n$ są literałami. Jeśli $n = 0$, to klauzulę nazywamy klauzulą pustą i oznaczamy symbolem $\perp$.

Postać klauzulowa:

$$\forall \overline{x_1}\ C_1 \wedge \ldots \wedge \forall \overline{x_m}\ C_m \qquad (m \geq 1)$$

gdzie $C_1, \ldots, C_n$ są klauzulami, natomiast $\overline{x_i}$ jest listą wszystkich zmiennych występujących w C_i.

Rezolucja – rezolwenta binarna

Niech C_1 i C_2 będą klauzulami bez wspólnych zmiennych.
Niech $l_1, \ldots, l_n \in C_1$ $(n > 0)$ i $l'_1, \ldots, l'_k \in C_2$ $(k > 0)$.
Załóżmy, że wszystkie literały $l_1, \ldots, l_n$ są dodatnie i wszystkie literały $l'_1, \ldots, l'_k$ są ujemne lub odwrotnie.

Jeśli θ jest najogólniejszym unifikatorem zbioru $\{|\ l_1\ |, \ldots, |\ l_n\ |, |\ l'_1\ |, \ldots, |\ l'_k\ |\}$, to klauzulę

$$(C_1\theta - \{l_1\theta, \ldots, l_n\theta\}) \vee (C_2\theta - \{l'_1\theta, \ldots, l'_k\theta\})$$

nazywamy **binarną rezolwentą** klauzul C_1 i C_2.

Zauważmy, że n nie musi być równe k. (Jest to bardziej skompaktowany zapis rezolucji uwzględniający faktoryzację.)

Rezolucja – reguła rezolucji

Rezolwentą klauzul C_1 i C_2, oznaczaną $RES(C_1, C_2)$, nazywamy dowolną binarną rezolwentę wariantu klauzuli C_1 i wariantu klauzuli C_2.

Regułą rezolucji nazywamy regułę wnioskowania

$$\frac{C_1, C_2}{RES(C_1, C_2)}.$$

Na przykład,

$$\frac{\neg Rich(x) \vee Unhappy(x), \qquad Rich(Ken)}{Unhappy(Ken)}$$

z unifikacją $\theta = \{x/Ken\}$

Rezolucja – dowód rezolucyjny

Przypomnienie idei: Stosujemy regułę rezolucji $KB \wedge \neg\alpha$ sprowadzonej do postaci klauzulowej

Dowód rezolucyjny ze zbioru klauzul CL:
Ciąg klauzul $C_1, \ldots, C_n$ taki, że $C_n = \perp$ i dla każdego $1 \leq i \leq n$, $C_i \in CL$ lub C_i jest rezolwentą C_j i C_k, gdzie $j, k < i$.

Twierdzenie
Niech CL będzie zbiorem klauzul nie zawierających symbolu "$=$". CL jest niespełnialny $\iff$ gdy istnieje dowód rezolucyjny z CL.

Aksjomaty równości

CL – zbiór klauzul.

$EQ(CL)$ – zbiór aksjomatów równości dla CL:

$$x = x$$

$$x \neq y \vee y = x$$

$$x \neq y \vee y \neq z \vee x = z$$

$$x_1 \neq y_1 \vee \cdots \vee x_n \neq y_n \vee \neg P(x_1, \ldots, x_n) \vee P(y_1, \ldots, y_n),$$

dla każdego n-argumentowego ($n > 0$) symbolu predykatowego P występującego w CL

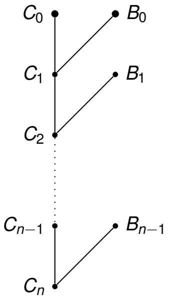
$$x_1 \neq y_1 \vee \cdots \vee x_n \neq y_n \vee (F(x_1, \ldots, x_n) = F(y_1, \ldots, y_n)),$$

dla każdego n-argumentowego ($n > 0$) symbolu funkcyjnego F występującego w CL.

Twierdzenie: Niech CL będzie zbiorem klauzul z równością. CL jest niespełnialny $\iff$ istnieje dowód rezolucyjny z $CL \cup EQ(CL)$.

Rezolucja liniowa

<u>Liniowym</u> dowodem rezolucyjnym ze zbioru klauzul CL nazywamy dowód postaci



gdzie:

1. $C_0 \in CL$
2. Dla $0 \leq i < n$, $B_i \in CL$ lub $B_i = C_j$, dla pewnego $j < i$
3. Dla $1 \leq i \leq n$, C_i jest rezolwentą C_{i-1} i B_{i-1}.
4. $C_n = \perp$.

Twierdzenie

Niech CL będzie zbiorem klauzul nie zawierających symbolu "$=$". CL jest niespełnialny $\iff$ istnieje liniowy dowód rezolucyjny z CL.

Twierdzenie

Niech CL będzie zbiorem klauzul z równością. CL jest niespełnialny $\iff$ istnieje liniowy dowód rezolucyjny ze zbioru $CL \cup EQ(CL)$.

Klauzulę C_0 w liniowym dowodzie rezolucyjnym nazywamy klauzulą główną.

Twierdzenie

Niech CL będzie niespełnialnym zbiorem klauzul i niech $CL' \subseteq CL$. Jeśli zbiór $CL - CL'$ jest spełnialny, to istnieje liniowy dowód rezolucyjny z CL (ewentualnie z $CL \cup EQ(CL)$) z klauzulą główną $C \in CL'$.

Rezolucja: przekształcanie do postaci klauzulowej

Twierdzenie

Istnieje algorytm, który dla dowolnej formuły A znajduje formułę B taką, że:

(1) B jest w postaci klauzulowej.

(2) A jest spełnialna $\iff$ gdy B jest spełnialna.

(A i B nie muszą być równoważne.)

Rezolucja: przekształcanie do postaci klauzulowej

Każdy, kto kocha wszystkie zwierzęta, jest kochany przez kogoś:

$$\forall x\ [\forall y\ Animal(y) \implies Loves(x,y)] \implies [\exists y\ Loves(y,x)]$$

1. Eliminacja równoważności i implikacji

$$\forall x\ [\neg \forall y\ \neg Animal(y) \vee Loves(x,y)] \vee [\exists y\ Loves(y,x)]$$

2. Przemieszczenie $\neg$ do środka: $\neg \forall x\ p \equiv \exists x\ \neg p, \quad \neg \exists x\ p \equiv \forall x\ \neg p$:

$$\forall x\ [\exists y\ \neg(\neg Animal(y) \vee Loves(x,y))] \vee [\exists y\ Loves(y,x)]$$
$$\forall x\ [\exists y\ \neg\neg Animal(y) \wedge \neg Loves(x,y)] \vee [\exists y\ Loves(y,x)]$$
$$\forall x\ [\exists y\ Animal(y) \wedge \neg Loves(x,y)] \vee [\exists y Loves(y,x)]$$

Rezolucja: przekształcanie do postaci klauzulowej

3. Przemianowanie zmiennych tak, żeby każdy kwantyfikator miał inną

$$\forall x\ [\exists y\ Animal(y) \wedge \neg Loves(x,y)] \vee [\exists z\ Loves(z,x)]$$

4. Skolemizacja – usunięcie kwantyfikatorów egzystencjalnych.
   Każda zmienna kwantyfikowana egzystencjalnie jest zastępowana przez tak zwaną funkcję Skolema (więcej na następnym slajdzie).

$$\forall x\ [Animal(F(x)) \wedge \neg Loves(x,F(x))] \vee Loves(G(x),x)$$

5. Usunięcie kwantyfikatorów uniwersalnych

$$[Animal(F(x)) \wedge \neg Loves(x,F(x))] \vee Loves(G(x),x)$$

6. Rozdzielenie $\wedge$ względem $\vee$

$$[Animal(F(x)) \vee Loves(G(x),x)] \wedge [\neg Loves(x,F(x)) \vee Loves(G(x),x)]$$

Więcej o skolemizacji

Powtarzamy poniższy proces, aż wszystkie kwantyfikatory $\exists$ zostaną usunięte

Bierzemy dowolną podformułę postaci $\exists y\ B(y)$.

Niech $x_1, \ldots, x_n$ ($n \geq 0$) będzie listą wszystkich zmiennych wolnych występujących w $\exists y\ B(y)$, które są uniwersalnie kwantyfikowane w pewnej nadformule formuły $\exists y\ B(y)$.

Jeśli $n = 0$, zastępujemy $\exists y\ B(y)$ przez $B(A)$, gdzie A jest nową stałą indywiduową.

Jeśli $n > 0$, zastępujemy $\exists y\ B(y)$ przez $B(F(x_1, \ldots, x_n))$, gdzie F jest nowym symbolem funkcji n-argumentowej.

Uwaga: Skolemizacja nie zachowuje równoważności.
(Ale zachowuje spełnialność.)

Przykład

$\neg American(West) \vee \neg Weapon(y) \vee \neg Sells(x,y,z) \vee \neg Hostile(z) \vee Criminal(x)$	$\neg Criminal(West)$
$American(West)$	$\neg American(West) \vee \neg Weapon(y) \vee \neg Sells(West,y,z) \vee \neg Hostile(z)$
$\neg Missile(x) \vee Weapon(x)$	$\neg Weapon(y) \vee \neg Sells(West,y,z) \vee \neg Hostile(z)$
$Missile(M1)$	$\neg Missile(y) \vee \neg Sells(West,y,z) \vee \neg Hostile(z)$
$\neg Missile(x) \vee \neg Owns(Nono,x) \vee Sells(West,x,Nono)$	$\neg Sells(West,M1,z) \vee \neg Hostile(z)$
$Missile(M1)$	$\neg Missile(M1) \vee \neg Owns(Nono,M1) \vee \neg Hostile(Nono)$
$Owns(Nono,M1)$	$\neg Owns(Nono,M1) \vee \neg Hostile(Nono)$
$\neg Enemy(x,America) \vee Hostile(x)$	$\neg Hostile(Nono)$
$Enemy(Nono,America)$	$\neg Enemy(Nono,America)$

$\square$

Dziękuję za uwagę!