Algorytmiczne aspekty teorii gier:
Wykład 7 i 8
Gry parzystości na grafach konfiguracji (*nieskończonych*)

09-04-2003
16-04-2003

Spis treści

1 Wyklad 7

Wykład prowadził dr hab. Igor Walukiewicz
Notatki przygotował Łukasz Chmielewski
<lc189379@zodiac.mimuw.edu.pl>

1.1 Gry parzystości - przypomnienie

$$\mathbf{G} = \langle Pos_E, Pos_A, Moves, rank : Pos \to \{1, \dots d\}\rangle$$

Ewa wygrywa rozgrywkę $p_0 p_1 p_2 \dots$ jeśli $limsup_{n\to\infty} rank(p_n)$ jest parzyste lub Adam nie może wykonać ruchu. Oznacza to, że największy z rank, który powtarza się nieskończenie często, jest parzysty. Ważne jest założenie, że funkcja rangi idzie w skończony zbiór.

1.2 Gry zadane przez Maszyny Turinga

1.2.1 Maszyna Turniga

$$M = \langle Q, \Sigma, B, q_0, \delta : Q \times \Sigma \to P(Q \times \Sigma \times (l, r)), F\rangle$$

Objaśnienia:

M - maszyna Turinga;

Q - stany;

Σ - alfabet maszyny;

B - blank;

q_0 - stan początkowy;

δ - funkcja przejścia (l - lewo, r - prawo);

F - zbiór stanów końcowych;

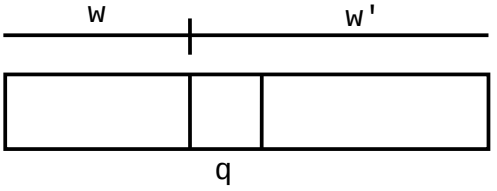
1.2.2 Konfiguracja Maszyny Turinga

$$wqw' \text{ gdzie } w, w' \in E^*$$

$$wqaw' \vdash wcq_1w' \Rightarrow (q_1, c, r) \in \delta(q, a)$$

$$wbqaw' \vdash wq_1bcw' \Rightarrow (q_1, c, l) \in \delta(q, a)$$

$$wq \vdash wcq_1 \Rightarrow (q_1, c, r) \in \delta(q, B)$$

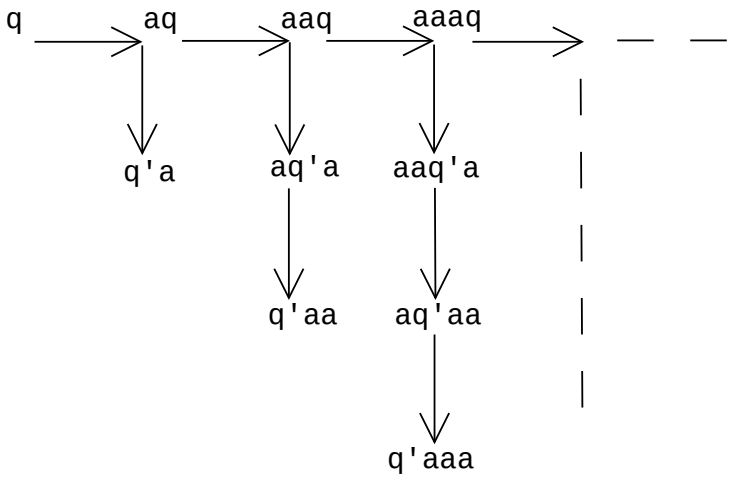


Rysunek 1: Konfiguracja

1.2.3 Graf

$$G(M) = \langle E^*QE^*, \vdash \rangle$$

To może być graf nieskończony, np.:



Rysunek 2: Graf

Dzielimy Q na Q_E i Q_A oraz bierzemy funkcję $\Omega : Q \to \mathbb{N}$. To zdefiniuje grę.

1.2.4 Gra

$$\mathbf{G}(M, Q_E, Q_A, \Omega) = \langle Pos_E = E^*Q_EE^*, Pos_A = E^*Q_AE^*, Moves = \vdash, rank \rangle$$

gdzie

$$rank(wqw') = \Omega(q)$$

Przykład:
Załóżmy, że M nie robi ruchów ze stanów F.

$$\begin{aligned} Q_E &= Q \setminus F \\ Q_A &= F \\ \Omega(q) &= 1 \\ \text{dla } q &\in Q \end{aligned}$$

Czyli: Adam przegra, gdy Ewa dojdzie do stanu akceptującego.

W $G(M, Q_E, Q_A, \Omega)$ Ewa ma strategię wygrywającą z konfiguracji $q_0 w$, wtedy i tylko wtedy, gdy $w \in L(M)$.

Fakt:

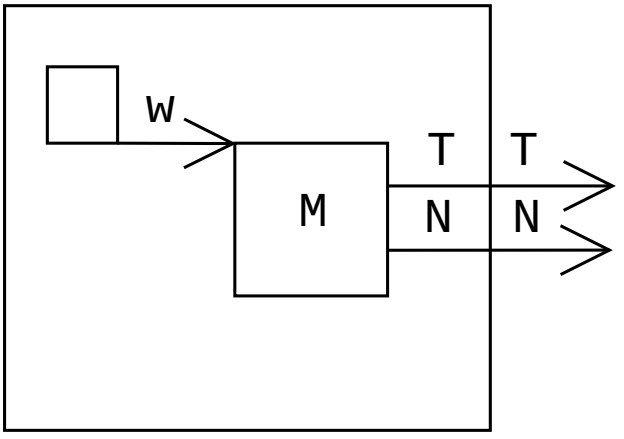
Problem rozwiązywania gier zadanych przez maszynę Turinga definiujejemy jako: dla danej gry i konfiguracji stwierdzić czy Ewa ma strategię wygrywającą.

Ponieważ problem : czy dane słowo należy do języka rozpoznawanego przez daną maszynę Turinga jest nierozstrzygalny to problem rozwiązywania gier zadanych przez maszynę Turinga jest również nierozstrzygalny (jak widać z przykładu).

Przykład:

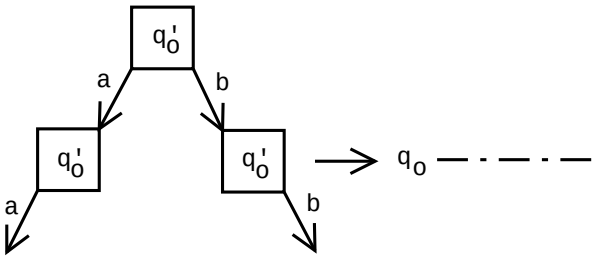
Bierzemy M jak poprzednio. Konstruujemy M'. Dodajemy do M gadżet generujący dowolne slowo w (rysunek 3).

$$\begin{aligned} wq_0' &\vdash wbq_0' \text{ dla dowolnego } b \in E \\ wq_0' &\vdash^* q_0 w \end{aligned}$$



Rysunek 3: w to dowolne wygenerowne słowo

$$G(M', Q_E = Q \setminus (F \cup \{q_0'\}), Q_A = (F \cup \{q_0'\}), \Omega)$$



Rysunek 4: Przykład drzewa gry

gdzie

$$\begin{aligned} \Omega(q_0') &= 0 \\ \Omega(q) &= 1 \text{ dla dowolnego } q \in Q \end{aligned}$$

Jak widać z drzewa gry (rysunek 4) Adam musi kiedyś wejść do q_0 (bo q_0' parzyste).

Ewa ma strategię z q_0' wtedy i tylko wtedy, gdy M akceptuje każde słowo.

Zatem problem rozwiązywania gier zadanych przez maszynę Turinga nierostrzygalny. Widać, że ten problem jest trudniejszy niż problem: czy dane słowo należy do języka rozpoznawanego przez daną maszynę Turinga (a już ten problem jest nieroztrzygalny).

1.3 Gry zadane przez Automaty ze stosem

1.3.1 Definicje

Automat ze stosem:

$$P = \langle Q, \Gamma, \delta : Q \times \Gamma \to P(Q \cup Q \times \Gamma) \rangle$$

Objaśnienia:

P - automat;

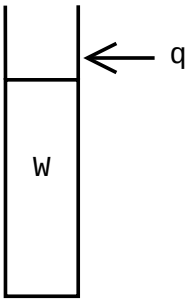
Q - stany;

Γ - alfabet automatu (zawiera alfabet słów i stosu, gdyż nie zajmujemy się działaniem automatu);

δ - funkcja przejścia;

Konfiguracja:

$$qw \text{ , gdzie } w \in \Gamma^*, \ q \in Q$$

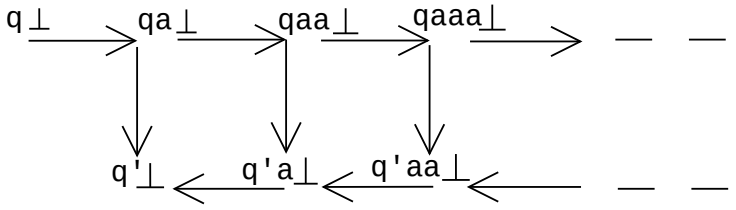


Rysunek 5: Stos automatu

Robimy założenie: $\perp \in \Gamma$ - litera, której nie można wkładać/zdejmować ze stosu.

$$\begin{aligned} qzw &\vdash q'w \text{ dla } q' \in \delta(q,z) \\ qzw &\vdash q'z'zw \text{ dla } (q',z') \in \delta(q,z) \end{aligned}$$

Przykład:



Rysunek 6: Graf dla automatu ze stosem

Funkcja prześcia dla przykładowego automatu (z rysunku 6):

$$\begin{aligned} \delta(q,a) &= (q,a) \\ \delta(q,\perp) &= (q,a) \\ \delta(q,a) &= q' \\ \delta(q',\perp) &= q' \end{aligned}$$

Grę definiujemy analogicznie jak dla maszyn Turing'a.

1.3.2 Twierdzenie

Problem rozwiązywania gier to:
Dla danych : P, Q_E, Q_A, Ω oraz kofiguracji qw stwierdzić, czy Ewa ma strategię wygrywającą z qw w $G(P, Q_E, Q_A, \Omega)$.

Problem rozwiązywania gier zadanych przez automat ze stosem jest $EXPTIME$*-zupełny*.

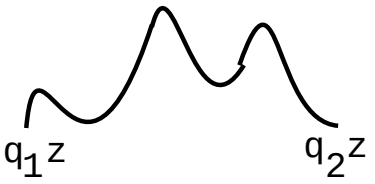
1.3.3 Warunek P1

Definiujemy grę z warunkiem **P1** następująco:
Nie ma przejść ze stanu Q_A, oraz $\Omega(q) = 1$, dla każdego $q \in Q$.

Twierdzenie:
Gry z warunkiem **P1** są rozwiązywalne w czasie wielomianowym.
Dowód:

1. Liczymy zbiór K trójek (rysunek 7) (q_1, z, q_2) takich, że istnieje obliczenie

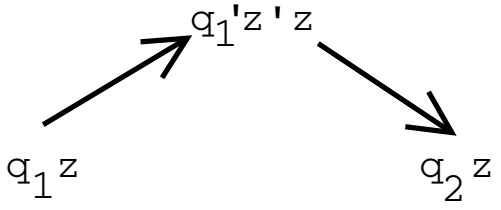
$$q_1 z \vdash^* q_2 z$$



Rysunek 7: przyklad

(a) Wstawiamy (q_1, z, q_2) do K (rysunek 8), gdy

$$q_1 z \vdash q_1' z' z \vdash q_2 z$$



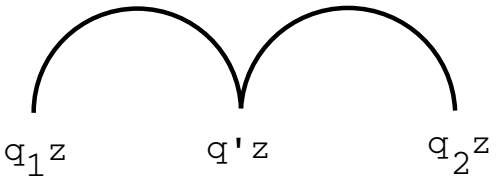
Rysunek 8: przyklad

(b) Wstawiamy (q_1, z, q_2) do K (rysunek 9), gdy

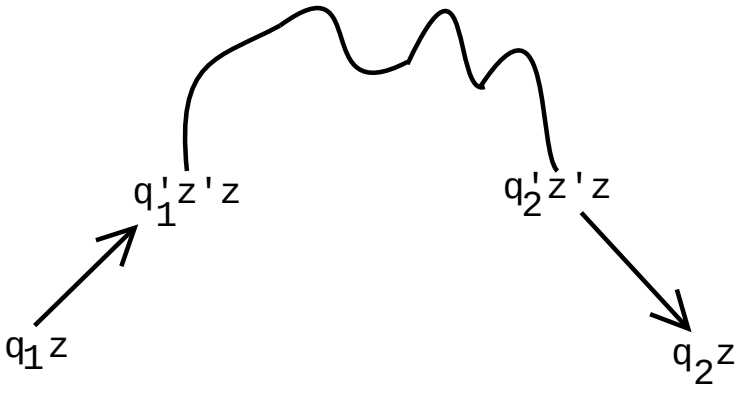
$$(q_1, z, q') \in K \text{ i } (q', z, q_2) \in K$$

(c) Wstawiamy (q_1, z, q_2) do K (rysunek 10), gdy

$$\begin{aligned} (q_1', z', q_2') &\in K \\ q_1 z &\vdash q_1' z' z \\ q_2' z' z &\vdash q_2 z \end{aligned}$$



Rysunek 9: przyklad



Rysunek 10: przyklad

2. **Lemat:**
   Zbiór K to najmniejszy zbiór zamknięty na powyższe operacje (a) i (b) i (c).

   Definiujemy nową grę G^* (rysunek 11):

$$G^* = \langle Pos_E = Q_E \times \Gamma, Pos_A = Q_A \times \Gamma, Moves, rank \rangle$$

   $Moves$:

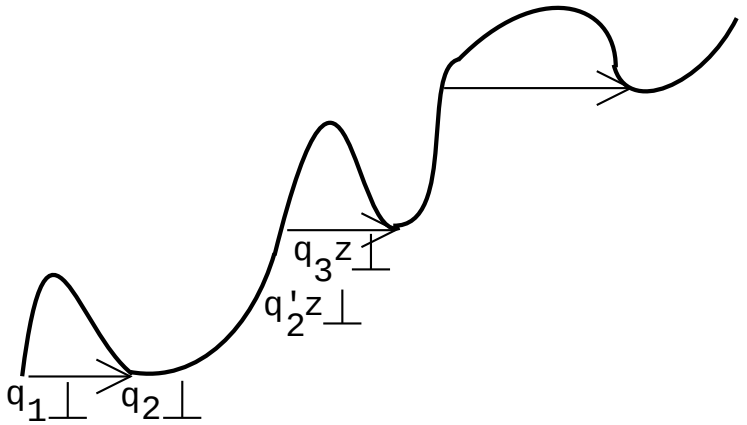
$$q_1 z \to q_2 z \text{ jeśli } q_1 z q_2 \in K$$
$$q_1 z \to q_2 z' \text{ jeśli } q_2 z \in \delta(q, z)$$

   $Rank$:

$$rank(q, z) = 1$$

3. **Lemat:**
   Ewa wygrywa z pozycji qz w $G(P, Q_E, Q_A, \Omega)$, wtedy i tylko wtedy, gdy Ewa wygrywa z qz w G^*.

   Dowód: indukcja po liczbie kroków do wykonania (długości ścieżki).



Rysunek 11: Nowa gra G*

W związku z konstrukcją nowej gry widać, że gra G^* jest rozwiązywalna w czasie wielomianowym od swojego rozmiaru. Zatem gra G jest również rozwiązywalna w czasie wielomianowym w stosunku do wielkości automatu.

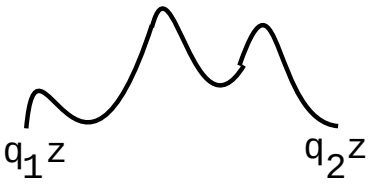
1.3.4 Warunek P2

Warunek **P2** jest formułowany analogicznie do **P1**.

$$\begin{aligned} Q_E &= Q \\ \Omega(q) &\in \{1,2\} \text{ , gdzie } q \in Q \end{aligned}$$

Twierdzenie:
Gry z warunkiem **P2** są rozwiązywalne w czasie wielomianowym.
Dowód:
Liczymy zbiór K czwórek (rysunek 12) (q_1, z, q_2, b) takich, gdzie $b = true$, jeśli w obliczeniu wystąpił stan z rangą 2 (w przyciwnym przypadku $b = false$).

W związku z konstrukcją nowej gry widać, że gra G^* jest rozwiązywalna w czasie wielomianowym w stosunku do swojego rozmiaru. Zatem gra G jest również rozwiązywalna w czasie wielomianowym w stosunku do wielkości automatu.



Rysunek 12: przykład

Definiujemy G^* (analogicznie jak w **P1**):

$$\begin{aligned} q_1 z &\rightarrow^b q_2 z, \text{ jeśli } (q_1, z, q_2, b) \in K \\ q_1 z &\rightarrow^b q_2 z', \text{ jeśli } q_2 z' \in \sigma(q, z) \text{ i } b \equiv (\Omega(q_1) = 2 \text{ lub } \Omega(q_2) = 2) \end{aligned}$$

Ewa wygrywa w G^* wtedy i tylko wtedy, gdy rozgrywka przechodzi nieskończenie często przez krawędź z $b = true$.

1.3.5 Warunek P3 - alternująca osiągalność

Warunek **P3** jest formułowany analogicznie do P1.

$$\Omega(q) = 1, q \in Q$$

Czyli w grach z tym warunkiem Adam „może się bronić".

Dla dowolnego G i V definiujemy odległość atraktorową zbioru wierzchołków w G przez $u \in Attr_E(V)$. Odległość u od V to najmniejsza τ taka, że $u \in Attrl_E^\tau(V)$.

Odległość wierzchołkowa = numer warstwy.

Liczone jest (analogicznie jak w **P1**):

$$(q, z, S) \text{ takie, że } S \subseteq Q$$

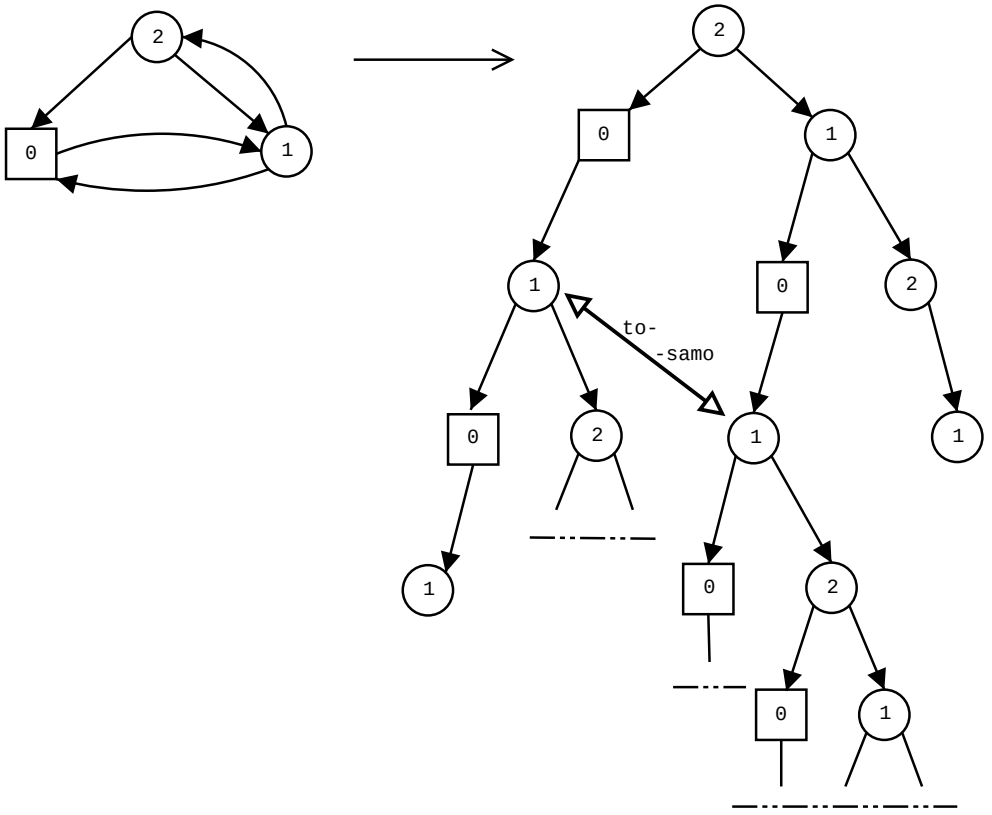
Kontynuacja na następnym wykładzie.

2 Wyklad 8

Wykład prowadził dr hab. Damian Niwiński
Notatki przygotował Michał Wełnicki
<mw189448@zodiac.mimuw.edu.pl>

2.1 Przypomnienie

2.1.1 Rozwinięcie grafu skończonego w nieskończony



2.1.2 Grafy istotnie i nieistotnie nieskończone

Drzewa nieskończone mające skończenie wiele nieizomorficznych poddrzew są nieistotnie nieskończone.

Drzewo z przykładu 2.1.1 jest więc *nieistotnie nieskończone*.

2.1.3 Definicja automatu

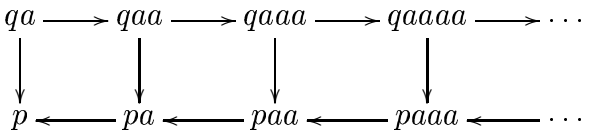
Automat ze stosem: $(Q, \Gamma, \delta : Q \times F \to \mathcal{P}(Q \cup (Q \times \Gamma^2)))$
Notacja

$qz \to q'\alpha$ (jeśli $q'\alpha \in \delta(q, z)$)

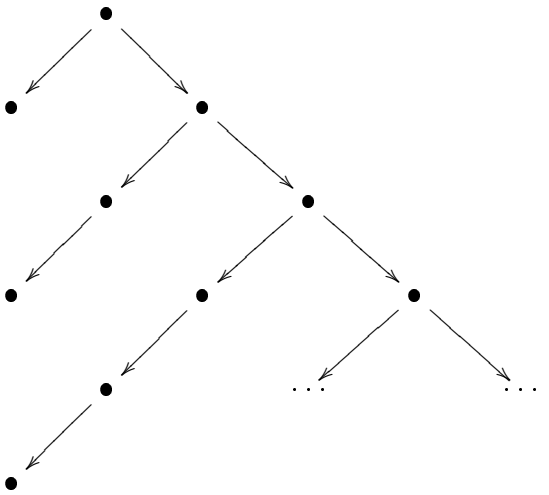
$qzw \vdash q'\alpha w$

2.1.4 Przykład rozwinięcia grafu konfiguracji

$qa \vdash qaa$
$qa \vdash p$
$pa \vdash p$



Rozwinięciem tego grafu konfiguracji jest następujące drzewo:



i jest to drzewo *istotnie nieskończone*.

2.2 Gra na grafie konfiguracji automatu ze stosem

2.2.1 Definicja gry

Grę otrzymujemy wprowadzając rozbicie $Q = Q_A \cup Q_E$ i $rank : Q \to \omega$
Rozważamy gry, gdzie $rank(Q) = \{1\}$ (gry na przetrwanie dla Adama).

2.2.2 Twierdzenie

Istnieje algorytm, który (w czasie wykładniczym) oblicza zbiory pozycji wygrywających w takich grach.

2.2.3 Definicja gry $G(q, z, S)$

Pomocniczo dla $q \in Q$, $z \in \Gamma$ i $S \subseteq Q$ określamy grę $G(q, z, S)$ z wyróżnioną pozycją startową qz i ruchach takich jak w grze G, z tym, że pozycja postaci q jest wygrywająca dla Ewy $\longleftrightarrow q \in S$

Pokażemy algorytm rozstrzygający, czy pozycja qz jest wygrywająca w grze $G(q, z, S)$. Żeby stwierdzić, czy pozycja qz jest wygrywająca w oryginalnej grze G wystarczy rozważyć $G(q, z, Q_A)$

2.2.4 Określamy grę skończoną G^*

Pozycje:

Pozycje „merytoryczne”

$Good(q, z, S)$ – pozycja Adama lub Ewy, w zależności od tego, czyj jest stan q

Pozycje pomocnicze:

T (true, wygrywa Ewa)

F (false, wygrywa Adam)

$Push(q, z, S, q', z')$ (pozycja Ewy)

$Choose(q, z, S, q', z', S')$ (pozycja Adama)

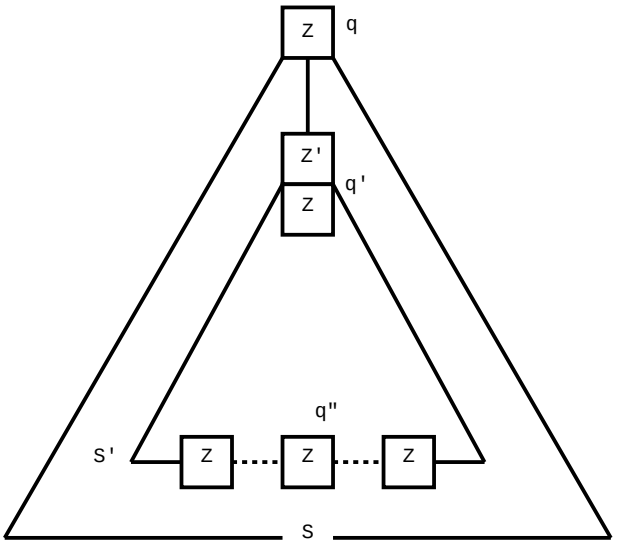
Ruchy:

$Good(q, z, S) \to$ T o ile $qz \vdash q' \in S$

$Good(q, z, S) \to$ F o ile $qz \vdash q' \notin S$

$Good(q, z, S) \to Push(q, z, S, q', z')$ o ile $qz \vdash q'z'z$

$Push(q, z, S, q', z') \to Choose(q, z, S, q', z', S')$, S' dowolne



$$Choose(q, z, S, q', z', S') \longrightarrow Good(q', z', S')$$

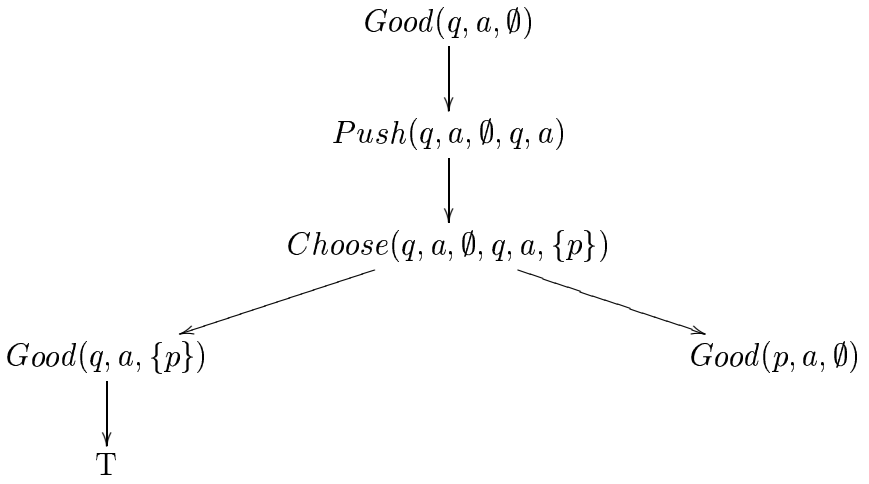
$$Good(q'', z, S) \qquad q'' \in S'$$

(rozszczepienie pozycji)

Przykład

Zdefiniujmy grę na grafie konfiguracji z przykładu 2.1.4, ustalając $Q_E = Q$.

Wtedy fragment gry G^* ma następującą postać:



2.2.5 Twierdzimy

Ewa wygrywa w G^* z pozycji $Good(q, z, S) \iff$ Ewa wygrywa $G(q, z, S)$ z pozycji qz.

Uwaga

Ponieważ gra G^* jest zwykłą grą skończoną, można ją rozwiązać w czasie liniowym. Jednak

liczba pozycji w G^* jest wykładnicza w stosunku do liczby stanów automatu skończonego. Stąd wynika wniosek, że oryginalną grę można rozwiązać w czasie *wykładniczym*.

Dowód twierdzenia 2.2.5

$\Longrightarrow$ Indukcja po $ord(Good(q, z, S))$ (odległości atraktorowej).

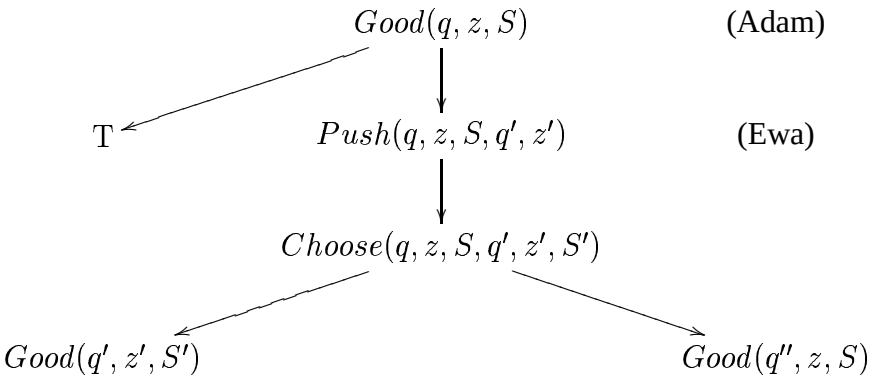
Jeśli $Good(q, z, S)$ jest w odległości 1 od T, to są dwa przypadki

(a) q Ewy $\quad Good(q, z, S) \to \mathrm{T}$

(b) q Adama, to każda strzałka $Good(q, z, S) \to \mathrm{T}$

Krok indukcyjny:

Przypuśćmy q – pozycja Adama:



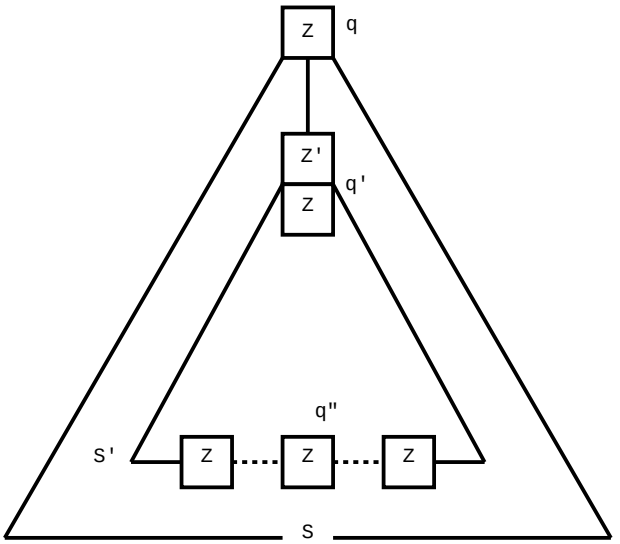
Z założenia indukcyjnego Ewa wygrywa $G(q', z', S')$ i $G(q'', z, S)$

$\Longleftarrow$ (dowód nie skończony)

Dla gry $G(q, z, S)$ określamy $Ord(q, z, S)$ jako $ord(q, z)$ w grze $G(q, z, S)$

Lemat o zmniejszaniu ord:

$$ord(q, z', S') < ord(q, z, S)$$


Z
q
Z'
q'
Z
q"
S'
Z
Z
Z
S