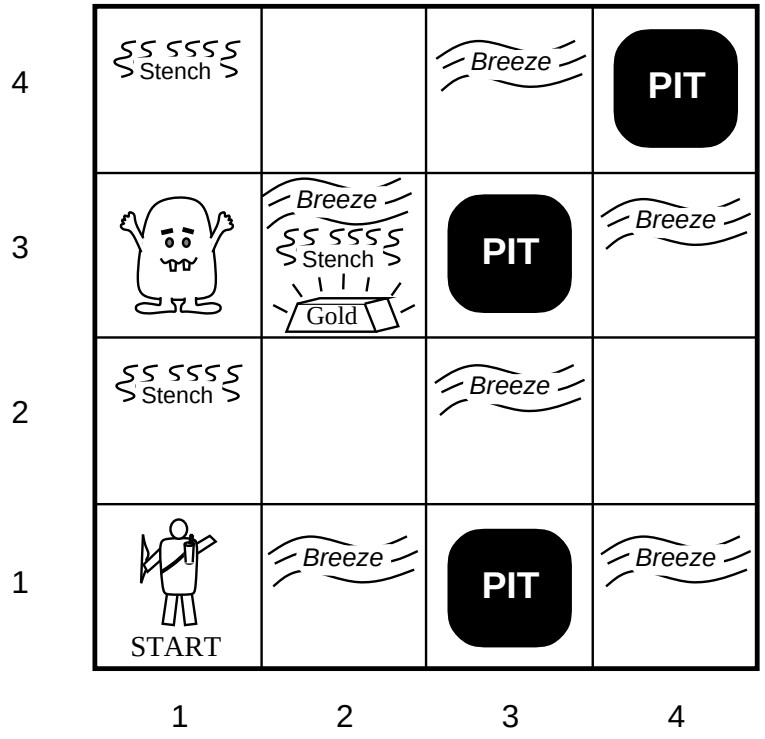
Świat Wumpusa (Wumpus world)



Obserwacje, akcje i cele agenta:

- W kwadracie gdzie mieszka Wumpus oraz w kwadratach ściśle przylegających agent czuje smród (stench).
- W kwadratach ściśle przylegających do jamy (pit), agent czuje wiatr (breeze).
- W kwadracie, gdzie znajduje się złoto, agent obserwuje błysk (glitter).
- Jeśli agent wejdzie na ścianę, czuje uderzenie (bump).
- Kiedy Wumpus zostaje zabity, w całej jaskini rozlega się wycie (scream).

- Obserwacje agenta reprezentujemy jako listę złożoną z 5 elementów.
  Przykładowo: [$Stench$, $Breeze$, $Glitter$, $None$, $None$].
  Agent nie potrafi rozpoznać kwadratu, w którym się znajduje.

- Akcje agenta: $Go-Forward$, $Turn-Left$, $Turn-Right$, $Grab$ (podnosi obiekt znajdujący się w tym samym kwadracie co agent),
  $Release$ (pozbywa się posiadanego przedmiotu),
  $Shoot$ (wypuszcza strzałę w kierunku, w którym patrzy; strzała albo zabija Wumpusa, albo odbija się od zewnętrznej ściany jaskini; agent ma tylko jedną strzałę),
  $Climb$ (opuszcza jaskinię, o ile znajduje się w kwadracie startowym).

- Agent ginie jeśli wejdzie do kwadratu, w którym znajduje się jama lub Wumpus.
- Celem agenta jest znalezienie złota i wydostanie się z jaskini. Dokładniej: 1000 punktów za wydostanie się z jaskini ze złotem, -1 punkt kary za każdą wykonaną akcję i -10000 punktów kary za zginięcie.

Wnioskowanie w świecie Wumpusa

1,4	2,4	3,4	4,4
1,3	2,3	3,3	4,3
1,2 **OK**	2,2	3,2	4,2
1,1 **A** **OK**	2,1 **OK**	3,1	4,1

(a)

A = *Agent*
B = *Breeze*
G = *Glitter, Gold*
OK = *Safe square*
P = *Pit*
S = *Stench*
V = *Visited*
W = *Wumpus*

1,4	2,4	3,4	4,4
1,3	2,3	3,3	4,3
1,2 **OK**	2,2 **P?**	3,2	4,2
1,1 **V** **OK**	2,1 **A** **B** **OK**	3,1 **P?**	4,1

(b)

1,4	2,4	3,4	4,4
1,3 **W!**	2,3	3,3	4,3
1,2 **A** **S** **OK**	2,2 **OK**	3,2	4,2
1,1 **V** **OK**	2,1 **B** **V** **OK**	3,1 **P!**	4,1

(a)

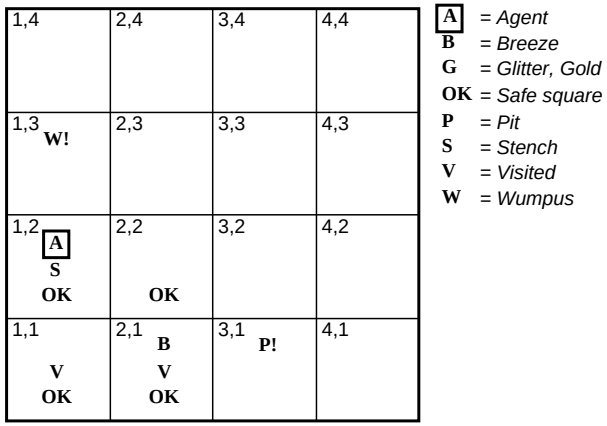
A = *Agent*
B = *Breeze*
G = *Glitter, Gold*
OK = *Safe square*
P = *Pit*
S = *Stench*
V = *Visited*
W = *Wumpus*

1,4	2,4 **P?**	3,4	4,4
1,3 **W!**	2,3 **A** **S** **G** **B**	3,3 **P?**	4,3
1,2 **S** **V** **OK**	2,2 **V** **OK**	3,2	4,2
1,1 **V** **OK**	2,1 **B** **V** **OK**	3,1 **P!**	4,1

(b)

Jak sformalizować świat Wumpusa w rachunku zdań?

Wiedza agenta po wykonaniu trzeciego ruchu. Bieżąca lista obserwacji: [$Stench$, $None$, $None$, $None$, $None$].



Obserwacje:

$$\neg S_{1,1},\ \neg B_{1,1},\ \neg S_{2,1},\ B_{2,1},\ S_{1,2},\ \neg B_{1,2}.$$

Wiedza o środowisku:

$$\neg S_{1,1} \Rightarrow \neg W_{1,1} \wedge \neg W_{1,2} \wedge \neg W_{2,1} \quad (1)$$
$$\neg S_{2,1} \Rightarrow \neg W_{1,1} \wedge \neg W_{2,1} \wedge \neg W_{2,2} \wedge \neg W_{3,1} \quad (2)$$
$$\neg S_{1,2} \Rightarrow \neg W_{1,1} \wedge \neg W_{1,2} \wedge \neg W_{2,2} \wedge \neg W_{1,3} \quad (3)$$
$$S_{1,2} \Rightarrow W_{1,3} \vee W_{1,2} \vee W_{2,2} \vee W_{1,1} \quad (4)$$

KB – koniunkcja obserwacji i formuł (1) - (4).

Chcemy pokazać, że z KB wynika $W_{1,3}$. W tym celu wystarczy pokazać, że formuła

$$KB \Rightarrow W_{1,3} \quad (5)$$

jest tautologią. Metoda zero-jedynkowa jest niepraktyczna (4096 wierszy). Łatwo wykazać, że (5) jest tautologią, posługując się regułą odrywania, eliminacji koniunkcji i rezolucji.

Uwaga: Rachunek zdań nie jest wygodny do formalizacji świata Wumpusa (zbyt wiele symboli zdaniowych, co czyni formalizację nieefektywną).

Jak reprezentować akcje?

$$A_{1,1} \wedge East_A \wedge W_{2,1} \Rightarrow \neg Forward.$$

Jeśli używamy rachunku zdań, nie możemy zadać bazie wiedzy pytania "Jaką akcję wybrać?" Musimy zadać serię pytań, każde dotyczące innej akcji: "Czy mam iść naprzód", Czy mam się odwrócić w prawo?", itp.

Model agenta posługującego się bazą wiedzy w rachunku zdań:

```
function PROPOSITIONAL-KB-AGENT(percept) returns an action
  static: KB, a knowledge base
          t, a counter, initially 0, indicating time

  TELL(KB, MAKE-PERCEPT-SENTENCE(percept, t))
  for each action in the list of possible actions do
      if ASK(KB, MAKE-ACTION-QUERY(t, action)) then
          t ← t + 1
          return action
  end
```

Świat Wumpusa zmienia się wraz z upływem czasu. Jak reprezentować te zmiany?

Jedyną metodą jest odpowiednie indeksowanie stałych zdaniowych:

$$
\begin{aligned}
A^0_{1,1} \wedge East^0_A \wedge W^0_{2,1} &\Rightarrow \neg Forward^0 \\
A^1_{1,1} \wedge East^1_A \wedge W^1_{2,1} &\Rightarrow \neg Forward^1 \\
A^2_{1,1} \wedge East^2_A \wedge W^2_{2,1} &\Rightarrow \neg Forward^2 \\
&\vdots \\
A^0_{1,1} \wedge North^0_A \wedge W^0_{1,2} &\Rightarrow \neg Forward^0 \\
A^1_{1,1} \wedge North^1_A \wedge W^1_{1,2} &\Rightarrow \neg Forward^1 \\
A^2_{1,1} \wedge North^2_A \wedge W^2_{1,2} &\Rightarrow \neg Forward^2 \\
&\vdots
\end{aligned}
$$

Jeśli przyjmiemy, że agent może wykonać 100 akcji w trakcie gry, to potrzebujemy 6400 formuł na wyrażenie prostego faktu, że nie należy iść do kwadratu, w którym znajduje się Wumpus.

Świat Wumpusa w logice pierwszego rzędu

Typowa obserwacja:

$$Percept([Stench, Breeze, Glitter, None, None], 5).$$

Akcje: $Turn(Right)$, $Turn(Left)$, $Forward$, $Grab$, $Shoot$, $Release$, $Climb$.

Konstruuje się pytanie typu: $\exists a\ Action(a, 5)$.

Prawidłowa odpowiedź: $\{a/Grab\}$.

Prosty agent reaktywny

$$\forall s, b, u, c, t \; Percept([s, b, Glitter, u, c], t) \Rightarrow Action(Grab, t).$$

Lepiej:

$$\forall s, b, u, c, t \; Percept([s, b, Glitter, u, c], t) \Rightarrow AtGold(t)$$

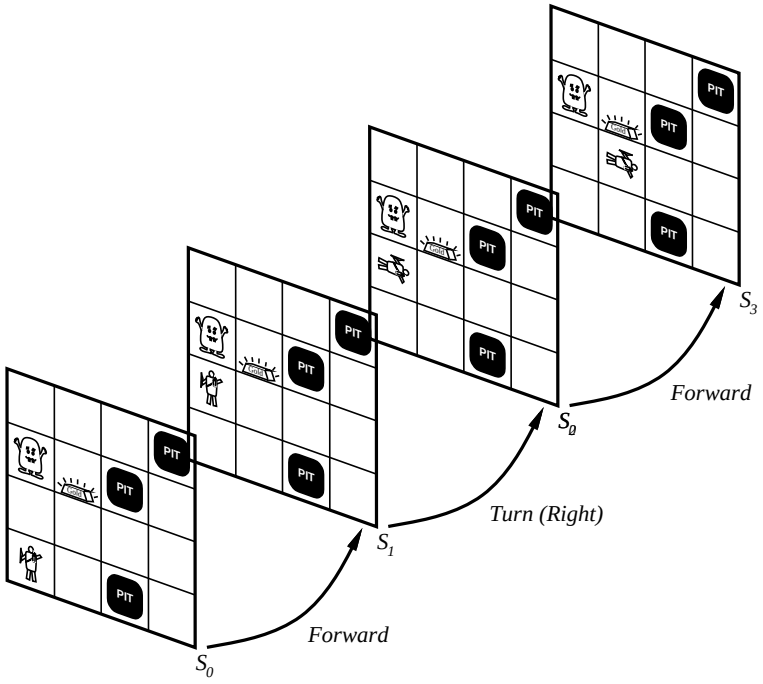
$$\forall t \; AtGold(t) \Rightarrow Action(Grab, t).$$

Prosty agent reaktywny nie poradzi sobie ze światem Wumpusa: Nigdy nie będzie wiedział, jak wrócić do kwadratu startowego w celu wykonania akcji $Climb$; często się zapętli.

Agent reaktywny

Agent reaktywny reaguje na bieżącą obserwację w oparciu o aktualny stan świata. Baza wiedzy takiego agenta musi więc mieć charakter dynamiczny.

Formalizacja w logice pierwszego rzędu opiera się na **Rachunku sytuacyjnym**.



Każdy symbol reprezentujący relację zmieniającą się w czasie, posiada dodatkowy argument, odpowiadający sytuacji:

$$At(Agent, [1, 1], S_0) \wedge At(Agent, [1, 2], S_1).$$

Definiowanie akcji

Efekty.

$Portable(Gold)$.

$\forall s\ AtGold(s) \Rightarrow Present(Gold, s)$.

$\forall x, s\ Present(x, s) \wedge Portable(x) \Rightarrow Holding(x, Result(Grab, s))$.

$\forall x, s\ \neg Holding(x, Result(Release, s))$.

Aksjomaty tła (frame axioms):
(1) Jeśli agent posiada przedmiot i go nie wyrzuca, to posiada ten przedmiot w następnym stanie.

$\forall a, x, s\ Holding(x, s) \wedge (a \neq Release) \Rightarrow Holding(x, Result(a, s))$.

(2) Jeśli agent nie posiada przedmiotu i go nie podnosi, to nie posiada go w następnym stanie.

$$\forall a, x, s\ \neg Holding(x, s) \wedge [a \neq Grab \vee \neg(Present(x, s) \wedge Portable(x))] \Rightarrow \neg Holding(x, Result(a, s)).$$

Aksjomaty definiujące efekty akcji można połączyć z aksjomatami tła, otrzymując

$$\forall a, x, s\ Holding(x, Result(a, s)) \Leftrightarrow [a = Grab \wedge Present(x, s) \wedge Portable(x)) \vee (Holding(x, s) \wedge a \neq Release)].$$

Powyższy aksjomat nazywamy **aksjomatem następnika stanu (successor-state axiom)**.

Liczba aksjomatów następnika stanu = liczba symboli predykatów zmieniających się w czasie.

Położenie agenta

W sytuacji początkowej: $At(Agent, [1, 1], S_0)$.

Co agent powinien wiedzieć o swojej pozycji?

1. Kierunek, w którym patrzy (0 – wschód, 90 – północ, 180 – zachód, 270 – południe).

   $Orientation(Agent, S_0) = 90$.

2. "Mapa" terenu. Mapę określa funkcja $LocationToward$, w skrócie LT, która dla danej pozycji i kierunku zwraca pozycję znajdującą się o jedną kratkę do przodu względem kierunku.

   $\forall x, y\ LT([x, y], 0) = [x + 1, y]$.
   $\forall x, y\ LT([x, y], 90) = [x, y + 1]$.
   $\forall x, y\ LT([x, y], 180) = [x - 1, y]$.
   $\forall x, y\ LT([x, y], 270) = [x, y - 1]$.

Posługując się funkcją LT, możemy określić kwadrat znajdujący się bezpośrednio przed agentem p:

$$\forall p, l, s\ At(p, l, s) \Rightarrow [LocationAhead(p, s) = LT(l, Orientation(p, s))].$$

Możemy też określić "ścisłe przyleganie":

$$\forall l_1, l_2\ Adjacent(l_1, l_2) \Leftrightarrow \exists d\ l_1 = LT(l_2, d).$$

3. "Geografia" terenu.

$$\forall x, y\ Wall([x, y]) \Leftrightarrow (x = 0 \vee x = 5 \vee y = 0 \vee y = 5).$$

4. Związek pomiędzy akcjami i położeniem agenta:

$$\forall a, l, p, s. At(p, l, Result(a, s)) \Leftrightarrow [(a = Forward \wedge l = LocationAhead(p, s) \wedge \neg Wall(l)) \vee (At(p, l, s) \wedge a \neq Forward)].$$

Własności świata

Akjomaty obserwacyjne: Opisują własności świata na podstawie obserwacji.

$$\forall l, s\ At(Agent, l, s) \wedge Stench(s) \Rightarrow Smelly(l).$$

$$\forall l_1, s\ Smelly(l_1) \Rightarrow [\exists l_2\ At(Wumpus, l_2, s) \wedge (l_2 = l_1 \vee Adjacent(l_1, l_2))].$$

$$\forall x, y, g, u, c, s\ Percept([None, None, g, u, c], s) \wedge At(Agent, x, s) \wedge Adjacent(x, y) \Rightarrow OK(y).$$

Aksjomaty przyczynowe: Opisują, w jaki sposób własności świata wpływają na obserwacje.

$$\forall l_1, l_2, s\ At(Wumpus, l_1, s) \wedge Adjacent(l_1, l_2) \Rightarrow Smelly(l_2).$$

$$\forall l, s\ \neg At(Wumpus, l, s) \wedge \neg Pit(l) \Leftrightarrow OK(l).$$

Wybór akcji

Klasyfikujemy akcje:

- $Great$ – Wzięcie złota i wydostanie się z jaskini ze złotem.
- $Good$ – Wejście do bezpiecznego kwadratu, który nie był jeszcze odwiedzany.
- $Medium$ – Wejście do bezpiecznego kwadratu, który był już odwiedzany.
- $Risky$ – Wejście do kwadratu, w którym może być Wumpus lub jama.
- $Deadly$ – Wejście do kwadratu, w którym na pewno jest Wumpus lub jama.

.

Jak sformalizować wybór akcji? Wprowadzamy dwuargumentowy symbol predykatowy $Action$. $Action(a, s)$ interpretujemy: w stanie s wybierz akcję a.

$\forall a, s\ Great(a, s) \Rightarrow Action(a, s).$

$\forall a, s\ Good(a, s) \wedge [\neg\exists b\ Great(b, s)] \Rightarrow Action(a, s).$

$\forall a, s\ Medium(a, s) \wedge [\neg\exists b\ Great(b, s) \vee Good(b, s)] \Rightarrow$
$Action(a, s).$

$\forall a, s\ Risky(a, s) \wedge [\neg\exists b\ Great(b, s) \vee Good(b, s) \vee$
$Medium(b, s)] \Rightarrow Action(a, s).$

Agent realizujący cel

Agent reaktywny potrafi rozsądnie poruszać się po jaskini. W momencie znalezienia złota celem agenta jest jak najszybszy powrót do kwadratu startowego i wydostanie się z jaskini.

$$\forall s \; Holding(Gold, s) \Rightarrow GoalLocation([1,1], s).$$

Agent powinien znaleźć sekwencję akcji (plan) realizującą ten cel. Istnieją dwie techniki rozwiązania tego zadania:

- Przeszukiwanie
- Planowanie

Problemy związane z formalizowaniem dynamicznie zmieniającego się świata

- **Problem tła** (frame problem) – Jak efektywnie reprezentować inercję? Połączenie aksjomatów opisujących efekty akcji i aksjomatów tła w aksjomaty następnika stanu nie rozwiązuje problemu dla akcji niedeterministycznych.

- **Problem kwalifikacji** (qualification problem) – Działając w rzeczywistym świecie, prawie nigdy nie wiemy, czy akcja się powiedzie. Jak sobie z tym poradzić formalizując dynamicznie zmieniający się świat?

- **Problem ramifikacji** (ramification problem) – Jak sformalizować uboczne efekty akcji?

Inżynieria wiedzy
(Knowledge engineering)

Zajmuje się metodologią konstruowania bazy wiedzy.

Inżynieria oprogramowania:

- Języki programowania.
- Programy.
- Kompilatory.
- Realizacja programów.

Inżynieria wiedzy:

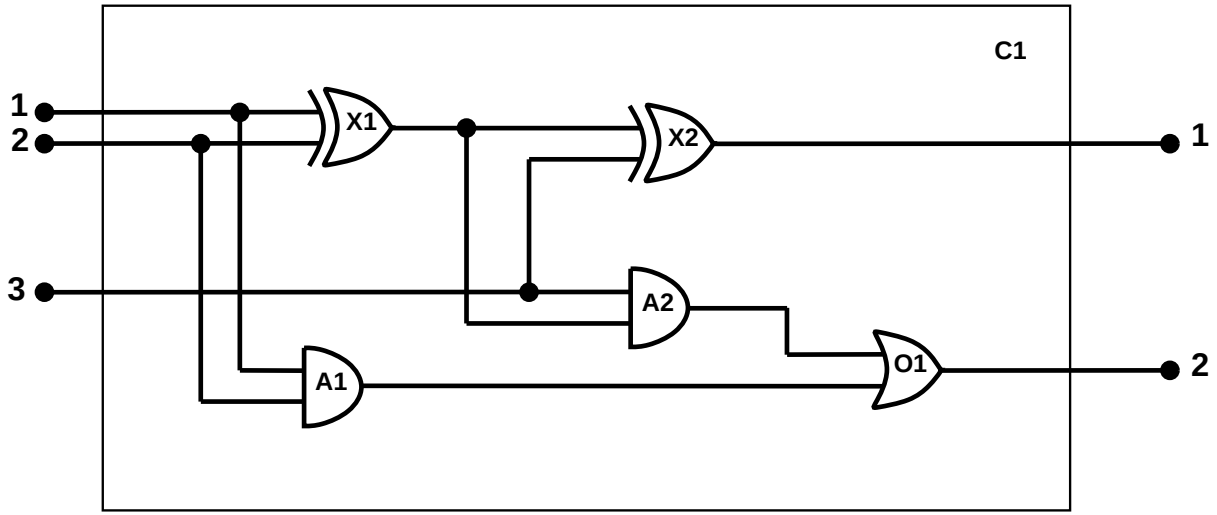
- Języki reprezentacji wiedzy.
- Bazy wiedzy.
- Procedury dowodzenia.
- Dowodzenie.

Metodologia konstruowania bazy wiedzy

1. Analiza dziedziny.
2. Wybór alfabetu.
3. Opis ogólnej wiedzy o dziedzinie.
4. Opis problemu.
5. Testowanie.

Uwaga: Nie należy oczekiwać, że skonstruowana baza wiedzy będzie od razu poprawna i pełna.

Przykład: prosty sumator



1. Analiza problemu.

- Ważne: bramki, wejście-wyjście, połączenia.
- Nieważne: przewody, wielkość, kolor.

2. Alfabet

- **Stałe**. Bramki: $X_1, X_2, A_1, A_2, O1$.
  Rodzaje bramek: XOR, OR, AND, NOT.
  Możliwe sygnały: On i Off.

- **Symbole funkcyjne**. $Type$ (zwraca rodzaj bramki), In i Out (zwracają wejścia i wyjścia (*terminale*) bramek, np. $In(1, X_1)$), $Signal$ (zwraca sygnał terminala).

- **Symbole predykatowe**. $Connected$ (zachodzi, jeśli dwa terminale są połączone).

Uwaga: To nie jest jedyny możliwy alfabet, który może być użyty do poprawnej formalizacji prostego sumatora.

3. Opis ogólnej wiedzy o dziedzinie (C – skrót od $Connected$.)

1. $\forall t_1, t_2\ C(t_1, t_2) \Rightarrow Signal(t_1) = Signal(t_2)$.
2. $\forall t\ Signal(t) = On \vee Signal(t) = Off$.
3. $On \neq Off$.
4. $\forall t_1, t_2\ C(t_1, t_2) \Leftrightarrow C(t_2, t_1)$.
5. $\forall g\ Type(g) = OR \Rightarrow [Signal(Out(1, g)) = On \Leftrightarrow$
   $\exists n\ Signal(In(n, g)) = On]$.
6. $\forall g\ Type(g) = AND \Rightarrow\ [Signal(Out(1, g)) = Off \Leftrightarrow$
   $\exists n\ Signal(In(n, g)) = Off]$.
7. $\forall g\ Type(g) = XOR \Rightarrow [Signal(Out(1, g)) = On \Leftrightarrow$
   $Signal(In(1, g)) \neq Signal(In(2, g))]$.
8. $\forall g\ Type(g) = NOT \Rightarrow [Signal(Out(1, g)) \neq Signal(In(1, g))]$.

4. Opis problemu

$$
\begin{aligned}
&Type(X_1) = XOR\\
&Type(X_2) = XOR\\
&Type(A_1) = AND\\
&Type(A_2) = AND\\
&Type(O_1) = OR\\
&C(Out(1, X_1), In(1, X_2))\\
&C(In(1, C_1), In(1, X_1))\\
&\vdots\\
&C(Out(1, O_1), Out(2, C_1))\\
&C(In(3, C_1), In(1, A_2)).
\end{aligned}
$$

5. Testowanie

Pytanie:

$$\begin{aligned}\exists i_1, i_2, i_3\ (&Signal(In(1, C_1)) = i_1)\\ &\wedge(Signal(In(2, C_1)) = i_2)\\ &\wedge(Signal(In(3, C_1)) = i_3)\\ &\wedge(Signal(Out(1, C_1)) = Off)\\ &\wedge(Signal(Out(2, C_1)) = On).\end{aligned}$$

Poprawna odpowiedź:

$$\begin{aligned}&(i_1 = On \wedge i_2 = On \wedge i_3 = Off)\vee\\ &(i_1 = On \wedge i_2 = Off \wedge i_3 = On)\vee\\ &(i_1 = Off \wedge i_2 = On \wedge i_3 = On).\end{aligned}$$