УДК 681.3.06

О.М. Цимбал

Харківський національний університет радіоелектроніки, Харків

ЛОГІЧНА МОДЕЛЬ АДАПТИВНОГО ПРИЙНЯТТЯ РІШЕНЬ

В статті пропонується метод адаптивного прийняття рішень для інтелектуальних системах керування роботами. Використання адаптивного прийняття рішень дозволятиме враховувати зміни у середовищі прийняття рішення. Розглядається логічна модель, що описує адаптивне прийняття рішень у роботизованих системах. У якості прикладів подання інформації використано мову програмування Пролог.

Ключові слова: *адаптивне прийняття рішень, логічна модель, предикатна модель, робот.*

Вступ

Під час розробки Інтелектуальних систем керування (ІСК) для мобільних і маніпуляційних роботів важливим етапом є вибір моделей прийняття рішень із урахуванням конкретних умов робочого простору, можливостей моделі робота, характеристик його сенсорної системи та інших показників. Характер обраних моделей визначає і сутність методів прийняття рішень.

Підхід теорії предикатів будується перш за все на основі математичної логіки. І це – не випадково. Заплановані дії мають виглядати послідовними, перевіреними і, отже, логічними у звичайному розумінні. Ще однією рисою прийнятого рішення є його істинність у відповідності до поточних умов робочого простору, тобто рішення, що приймається у просторі S, є істинним у момент часу t.

Метою запропонованої статті полягає у формулюванні основних позицій методу адаптивного прийняття рішень, описі логічної (предикатної) моделі прийняття рішень для ІСК роботами.

1. Метод адаптивного прийняття рішень та його особливості

Планування рішень відбувається в умовах як реального часу, так і поза ним. Як правило, початкове рішення, приймається в умовах постановки завдання і із урахуванням початкового стану усіх об'єктів, що беруть участь у процесі. При цьому відбувається пошук операторних процедур ІСК, що в найбільшій мірі задовольняють початковим і поточним умовам прийняття рішень. Розроблений план відповідатиме моделі прийняття рішень, обраній на початковому етапі.

На стадії виконання рішень відбувається реалізація запропонованого плану. Вона здійснюється в реальному часі і може обмежуватися тривалістю діяльності інших об'єктів і факторами внутрішнього характеру (наприклад, обмеженим зарядом акумуляторних батарей мобільного робота). Відхилення умов експлуатації робота від запланованих, виникненням зовнішніх факторів, що суттєво впливають на реалізацію рішень вимагатиме адаптації прийнятого плану на рівні перебудови окремих частин плану (стосовно розв'язання окремих підзадач) або перебудови практично усього плану розв'язання поставленої задачі. Таким чином, метод адаптивного прийняття рішень має враховувати зміни робочого простору і адаптувати прийняття рішень відповідно до них.

Метод адаптивного прийняття рішень є описом сукупності прийомів та операцій, що застосовуються у інтелектуальних системах керування роботами у розв'язанні складних завдань практичної робототехніки. Запропонуємо основні принципи такого методу.

1. Метод передбачає отримання інформації про стан робочого середовища робота за допомогою сенсорної системи.
2. Мета системи має формулюватися як новий можливий стан ІСК.
3. Мета системи має розглядатися як набір дискретних складових – підцілей ІСК.
4. Для досягнення поставленої мети ІСК має визначити набір дій – план рішення), що дозволяє послідовно досягти усі підцілі, що складають загальну мету.
5. Під час визначення загальної мети та окремих цілей ІСК має враховуватися як стан самої ІСК, так і стан робочого простору.
6. Якщо вплив власного стану ІСК та робочого простору не дозволяють виконувати окремі дії щодо з реалізації запланованих підцілей у досягненні загальної мети системи, план рішення має бути зміненим (адаптованим) таким чином, щоб забезпечити перехід від поточного стану ІСК до цільового (загальної мети).
7. Якість вибору плану досягнення мети ІСК має оцінюватися відповідно до умов прийняття рішень, поточного та наступного станів ІСК та робочого середовища робота.

Наступним кроком має бути детальний розгляд взаємодії описаних елементів і, одночасно, самого процесу прийняття рішень.

2. Опис моделі адаптивного прийняття рішень на основі логіки предикатів

Підхід теорії предикатів будується перш за все на основі математичної логіки. І це – не випадково. Заплановані дії мають виглядати послідовними, перевіреними і, отже, логічними у звичайному розумінні. Ще однією рисою прийнятого рішення є його істинність у відповідності до поточних умов робочого простору, тобто рішення, що приймається у просторі S, є істинним у момент часу t.

Під час розгляду логічної моделі використаємо ті ж самі позначення, як і у теоретико-множинній моделі попереднього підрозділу. Нехай інтелектуальна система керування (ІСК) в процесі виконання принятого рішення забезпечує перетворення початкового стану $\text{state}(x_0^0, x_1^0, x_2^0, \ldots x_{n-1}^0)$ у певний цільовий стан $\text{state}(x_0^m, x_1^m, x_2^m, \ldots x_{n-1}^m)$ характеризується набором параметрів

Якщо система (робот і світ навколо нього) в початковий момент часу складає множину аргументів $x_0^0, \ldots, x_n^0$ і характеризується станом state_0, тоді розглядаючи дискретний процес прийняття рішень, який складається є окремих дій $\text{action}_0, \ldots, \text{action}_k$, можна вказати, що перехід з одного дискретного стану в інший є певним відношенням між об'єктами:

$$\text{state}(x_0^1, x_1^1, \ldots, x_n^1) \leftarrow \text{action}_0(\text{state}(x_0^0, x_1^0, \ldots, x_n^0)),$$

де state – відношення (предикат) , що характеризує стан системи в цілому, action(state) – що означає дію щодо переходу з одного стану в інший.

Вся діяльність щодо переходу з одного стану системи в інший (шляхом виконання списку рішень) є набором предикатів

$$\text{state}(X^1) \leftarrow \text{action}_0(\text{state}(X^0)),$$

$$\text{state}(X^{21}) \leftarrow \text{action}_1(\text{state}(X^1)),$$

$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots..$$

$$\text{state}(X^{n-1} = Y) \leftarrow \text{action}_{n-2}(\text{state}(X^{n-2})).$$

Таким чином, мета системи прийняття рішень – знайти відповідну кількість action_i, які задовольняли б умовам станів state_i системи.

Однак, в умовах динамічного стану світу робота можлива ситуація, коли на певному стані $\text{state}(X^{i-1})$ дія $\text{action}_{i-1}(\text{state}(X^{i-1}))$ не призводить до переходу системи у стан $\text{state}(X^i)$, тобто:

$$\text{state}(X^i) \neq \text{action}_{i-1}(\text{state}(X^{i-1})).$$

В цих умовах на перший погляд необхідно знайти такий предикат action, який задовольнив би умові виразу. Однак, на ділі кількість можливих реальних дій є обмеженою (на відміну від кількості станів), тому вихід полягатиме у знаходженні такої дискретної послідовності $\overrightarrow{\text{action}}$ (вектор предикатів), що задовольнятиме меті системи.

Таким чином, якщо існує світ об'єктів є множиною X і початковим станом є X^0, тоді для досягнення цільового стану Y необхідно запланувати послідовність дій, що відображається предикатами action, а стан системи – предикатом state:

$$\text{state}(X^0),$$

$$\text{state}(X^1) \leftarrow \text{action}_0(\text{state}(X^0)),$$

$$\text{state}(X^2) \leftarrow \text{action}_1(\text{state}(X^1)),$$

$$\ldots\ldots\ldots\ldots$$

$$\text{state}(Y) \leftarrow \text{action}_{n-1}(\text{state}(X^{n-1})).$$

Якщо на певному етапі і, стан X^i є недосяжним, тобто $\text{state}(X^i) \neq \text{action}_{i-1}(\text{state}(X^{i-1}))$, тоді адаптивна система прийняття рішень має генерувати нову послідовність предикатів дій $\overrightarrow{\text{action}}$, що задовольнятиме змінам у робочому просторі:

$$\text{state}(X^{i1}) \leftarrow \text{action}_{n-1}^1(\text{state}(X^{i-1})),$$

$$\text{state}(X^{i2}) \leftarrow \text{action}_{n-1}^2(\text{state}(X^{i-1})),$$

і так далі, до

$$\text{state}(Y) \leftarrow \text{action}_{n-1}^{m-1}(\text{state}(X^{n-1})).$$

Аналогічна ситуація виникатиме коли ІСК має інформацію про зміну мети системи. Це означатиме, що цільовий стан state(Y) замінюється на певний $\text{state}(Y^i)$. Знову-таки це викликатиме необхідність генерації нової послідовності предикатів-дій.

План рішення буде складатися з наборів $\{\text{action}_0, \text{action}_1, \ldots, \text{action}_n\}$, а повний план складатиметься з усіх планів, що були розроблені під час прийняття рішення. Адаптований план рішення буде виражатися записом:

$$\text{plan}^{\text{adaptive}}(Y) \leftarrow \text{action}_0(\text{state}(X^0),$$
$$\text{action}_1(\text{state}(X^1), \text{action}_{n-1}(\text{state}(X^{n-1}))).$$

Такий план є кінцевим рішенням ІСК.

Вкажемо і на момент оцінки дій, що заплановані ІСК у складі послідовності предикатів.

Як відомо, предикат має значення істинності, у класичному значенні – відображення n аргументів на значення істинності. Хоча, зважаючи на апарат нечіткої логіки слід зауважити і на можливість введення поняття нечіткого предиката [1-3]. Класичний же предикат має лише значення true і false. З цієї точки зору перехід ІСК з одного стану $\text{state}(X_{i-1})$ в стан $\text{state}(X_i)$ теж матиме істинне або хибне значення, тобто система переходить у новий стан або ні. Напевно, що і в ідеалізованому підході стан цілої системи важко передбачити, тим більше надати стану системи однозначну оцінку у вигляді бінарних значень «істинно» і «хибно». Скоріше, істинність або хибність стосуватимуться окремих характеристик системи. Відповідно до теорії предикатів світ робота можна описати як набір відношень між об'єктами такого світу, наприклад, is_a –

приналежність до типу об'єктів, is_at – знаходження одного об'єкта біля іншого, stands – знаходження об'єкта у певному стані, тощо.

Наприклад, is_at(robot, position(x, y)) означає наявність робота у вказаній точці (x, y).

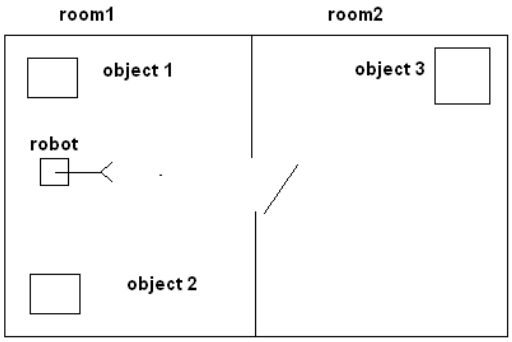

Рис. 1. Приклад простору робота

Для обмеженого світу робота (рисунок 1) трансформація станів системи під час прийняття рішень відображається станом поточної бази даних. Її статична частина має такий вигляд (у вигляді Пролог-програми):

is_a(room1, room, always)
is_a(room2, room, always)
is_a(room3, room, always)
is_a(box1, box, always)
is_a(box2, box, always)
is_a(box3, box, always).

Динамічна частина бази даних характеризує кожен з кроків роботи системи.

Стан 0:
is_at (object1, 10, 10)
is_at (robot, 10, 50)
is_at (object2, 10, 90)
stands (door12, closed)
is_at (object3, 120, 10).

Надалі, якщо ставиться завдання переміщення об'єкта 2 до об'єкта 3 і постає план виконання is_at(object2, object3). Відповідно до розробленого плану здійснюватиметься переміщення робота спочатку до об'єкта 2, потім (з ним) робот підходитиме до дверей 12, відкриватиме їх, і, насамкінець, переміщуватиметься до об'єкта 3. Все це відображається у послідовних змінах станів:

Стан 1:	***Стан 2:***
is_at (object1, 10, 10) is_at (robot, 10, 80) is_at (object2, 10, 90) stands (door12, closed) is_at (object3, 120, 10).	is_at (object1, 10, 10) is_at (robot, 50, 50) is_at (object2, 50, 50) stands (door12, closed) is_at (object3, 120, 10).
Стан 3:	***Стан 4:***
is_at (object1, 10, 10) is_at (robot, 70, 50) is_at (object2, 70, 50) stands (door12, opened) is_at (object3, 120, 10).	is_at (object1, 10, 10) is_at (robot, 110, 50) is_at (object2, 100, 50) stands (door12, opened) is_at (object3, 120, 10).

Таким чином, тільки частина фактів зазнає змін. Результат дій робота полягатиме у зміні наявності фактів певного роду. Додатково зазначимо, що у реальній Пролог-програмі необхідно розробити спеціальну процедуру, що реалізує генератор планів системи прийняття рішень. Метою генератора є пошук дій, здатних відповісти дискретним актам рішення, що розроблені АСПР. Більш детально, генератор планів розглядається у наступних розділах.

3. Логіка вищих порядків та її використання у системах прийняття рішень роботів

Сучасні мови програмування передбачають рівень взаємодії даних і процедур, що може забезпечувати керування обчислюваннями. Такі мови зазвичай базуються на функціональній парадигмі програмування і реалізовані процедури в них відповідають функціям, а об'єкти, що вводяться, мають тип високого порядку, як і функції, що забезпечують керування об'єктами. Таким чином, мова може йти про програмування високого порядку [4].

Аналогічні ідеї можна спрямувати і на логічне програмування. Зважаючи на те, що процедури у логічному програмуванні реалізуються предикатами, програмування високого порядку має відповідати включенню (інкапсуляції) предикатів у вирази на рівні термів. Попередні спроби, однак, виявилися невдалими з-за розгляду часткових випадків і відходом від класичного стилю логічного програмування.

У роботі [4] проводиться спроба реалізації методів програмування у термінах логіки високого порядку. Такий підхід має дати низку нових можливостей, серед яких, по-перше, модернізація мов логічного програмування до рівня сучасних мов функціонального типу; по-друге, використана часткова логіка на основі λ-термів, яка є базою для конструювання виразів предикатів; по-третє, використання логіки високого порядку розмиває різницю між предикатами і функціями, дозволяє проводити певного роду обчислення між предикатами і функціями, таким чином значно розширити можливості розвитку мови програмування. Самий термін «логіка високого порядку» має певні різні тлумачення:

1) філософи математики поділяють логіку на логіку першого та другого порядків, причому остання є формальною основою для усієї математики і, відповідно до теореми Геделя про неповноту, не може бути рекурсивно зведена до аксіом;

2) з точки зору доказів, будь-яка логіка система має бути формальною системою, і тут логіка високого порядку не є виключенням; основною різницею між логікою вищих порядків та логікою першого порядку є присутність у першій предикатних змінних і здатності формувати абстракції над логічними формулами;

3) логіка вищих порядків, як і будь-яка інша обчислювальна логіка містить λ-терми і змінні вищих порядків, включно з предикатними.

Власне, виникає запитання: яким чином можна і чи, взагалі, доречно використати логіку предикатів вищих порядків у системах прийняття рішень, чи не є це намаганням ускладнити процес прийняття рішень і який практичний зиск матиме розробник такої системи прийняття рішень?

Якщо пригадати наведені у цьому ж розділі моделі, процес прийняття рішень має своїм результатом певну послідовність дій, що їх має виконати робот задля поставленої мети. Цю послідовність можна зобразити, зокрема, так:

$$A_0 \rightarrow A_1 \rightarrow A_2 \rightarrow \ldots \rightarrow A_{n-1}.$$

Чим, однак, є вказані дії? Для мобільного робота ними будуть команди рухів, наприклад: вперед_100, праворуч_45, ліворуч_30; крім того, команди маніпуляцій: відкрити схват, закрити_схват. Окремо можуть використатися складені команди, типу розмістити об'єкт_А у позиції_N1, які в свою чергу розкладатимуться на команди здійснення рухів, маніпуляції, запити датчиків тощо. Лише окремі вказані команди є однорідними з точки зору побудови: вперед – назад, ліворуч – праворуч. Тим не менше, вони складають сутність операцій, виконуваних роботом, і в процесі виконання є атомарними виразами з точки зору логічної теорії [5, 6].

Питання про операції над об'єктами є одними з важливих з точки зору програмування. Якщо необ'єктно-орієнтовані мови виключають поняття класу, як опису об'єкта, що характеризується певними властивостями (змінними) та операціями (методами), то в мови, які підтримують концепцію об'єктно-орієнтованого програмування (ООП), реалізують концепцією класів – із властивими їм змінними-членами та функціями членами (методами), підтримують операції над об'єктами класів [7]. Для логічного програмування ситуація є іншою: окрім простих характеристик об'єктів, не виражених у явний спосіб, вводяться відношення між характеристиками, ці відношення позначаються предикатами. Проте, не у всіх реалізаціях мов логічного програмування можна передати предикат (навіть у формі факту), як аргумент іншого предиката, не можна додавати у базу даних правила, які вироблені у процесі виведення. Це, безумовно накладає певні обмеження на реалізацію програмного забезпечення.

З точки зору розробки систем прийняття рішень припущення «рівноправності» різних операцій робота має значно спрощувати весь процес. Наявність навіть простих операцій, наприклад, додавання та віднімання операцій, реалізованих між предикатами-діями робитиме процес прийняття рішень більш простим, як у сенсі наближення процесу до природного аналогу (людини-оператора), так і для планування процесу в автоматичному режимі. Тому слід звернути увагу на можливість опису процесів прийняття рішень роботів у термінах логіки предикатів вищих порядків.

Відмінність концепції предикатів високого порядку буде полягати у можливості проводити операції над окремими предикатами-діями, тобто проводити операції вигляду:

$$A_{i+1} = A_{i-1}(Op_i)A_i,$$

де Op_i - знак існування операції між двома предикатами, тим самим формулюючи план дій у більш прийнятний спосіб.

Наприклад, нехай існує ІКС, у якій мобільний робот R може виконувати дії маніпуляційного або локомоційного характеру. Тоді властивістю робота є набір дій (предикатів) $A = \{A_0, A_1, A_2 \ \ldots A_{n-1}\}$.

Для сумісного виконання дій між ними можуть вводитися операції:

$$Op = \{Op_0, Op_1, Op_2 \ \ldots Op_{n-1}\},$$

що можуть застосовуватися роботом для виконання завдання. Якщо початковий стан системи описати за аналогією попередніх розділів як $X^0 = \{X_0^0, \ldots, X_n^0\}$, а мету системи, як стан Y, тоді кожен предикат виконуватиме перетворення станів $X_{i+1} \leftarrow A(X_i)$, послідовно переводячи систему з початкового стану у цільовий, формуючи послідовність операцій виконання завдання.

Розглянемо приклад плану дій мобільного робота стосовно об'їзду перешкоди, зображеної на рис. 2.

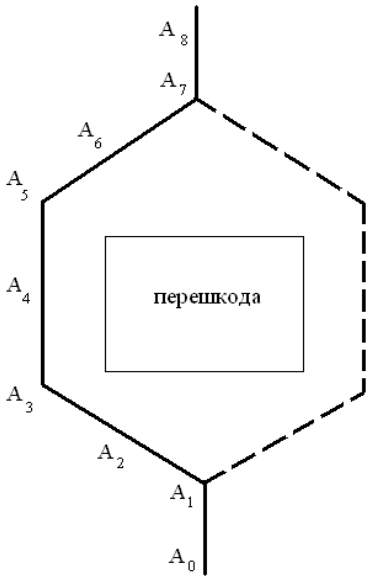

Рис. 2. Опис набору рухів мобільного робота для об'їзду перешкоди

Як можна припустити, прикладом рішення щодо об'їзду перешкоди може бути такий набір дій:

- рухатися (прямо, 100) (A_0);
- повернути (ліворуч, 60) (A_1);
- рухатися(прямо, 200) (A_2);
- повернути(праворуч, 60) (A_3);

– рухатися(прямо, 200) (A_4);
– повернути(праворуч, 60) (A_5);
– рухатися(прямо, 200) (A_6);
– повернути(ліворуч, 60) (A_7);
– рухатися(прямо, 100) (A_8).

Тобто вся послідовність дій щодо прийняття рішень матиме вигляд:

$A_0 + A_1 + A_2 + A_3 + A_4 + A_5 + A_6 + A_7 + A_8$,

причому результатом кожної операції буде зміна стану системи, що полягатиме у змінах координат робота. Напевно, якщо в ході роботи знадобиться обійти раптово виникаючу перешкоду, виникатиме потреба не тільки у операціях додавання, а й у відніманні дій робота, наприклад -рухатися(прямо, 100), хоча і тут можливими є варіанти рухатися(прямо, -100) або рухатися(назад, 100).

З точки зору мов програмування слід зауважити на те, що, наприклад, мова С++ не може просто складати функції (процедури) у певний набір. У мові Prolog послідовність виконання предикатів можна об'єднати у кон'юнктивний набір, але немає можливості проводити операції над предикатами і отримувати нові з точки зору прийняття рішень результати.

Використання логіки предикатів вищих порядків [8, 9] має суттєво спростити опис процесів прийняття рішень у робототехнічних системах, підвищити їх ефективність та гнучкість в умовах адаптації процесу прийняття рішень до змін у робочому просторі робота, змін у характеристиках робото технічної системи.

Висновки

Прийняття рішень у інтелектуальних системах керування роботами має враховувати багатостадійність розв'язання завдань маніпуляції об'єктами або навігації для мобільних систем. Системи прийняття рішень мають сприймати зміни середовища, у якому функціонує робот і продукувати такі схеми розв'язання завдань, які б відповідали змінам у середовищі функціонування. Зазначений підхід вимагає появи нової властивості систем прийняття рішень роботів, а саме – здатності гнучко перебудовувати (адаптувати) роботу у залежності від змін у навколишньому середовищі, змін у постановці мети або окремих підцілей, у стані самої робототехнічної системи. Формалізація такого підходу можлива в межах реалізації методу так окремих моделей адаптивного прийняття рішень.

Запропоновані метод і модель є основою для розробки інформаційного та програмного забезпечення інтелектуальних систем керування маніпуляційних та мобільних роботів різного призначення, що реалізовані автором мовами Пролог та С++.

Список літератури

1. Борисов, А.Н. Обработка нечеткой информации в системах принятия решений [Текст] / А.Н. Борисов, А.В. Алексеев, Г.В. Меркурьева. – М.: Радио и связь, 1989. – 304 с.

2. Hájek, Petr. Basic Fuzzy logic and BL-algebras [Текст] / Petr Hájek // Soft Computting. – 1998. – No. 2. – P. 124-128.

3. Lee, Richard C.T. Fuzzy logic and Resolution Principle / Richard C.T. Lee [Текст] // Journal of Association for Computing Machinery. – 1972. – Vol.19, No. 1. – P. 109-119.

4. *Nadathur, Gopalan. Higher-Order Logic Programming [Текст] / Gopalan Nadathur, Dale Miller // Handbook of Logic in AI and Logic Programming. Volume 5, Oxford University Press, 1998. – P. 499-590.*

5. Асаи, К. Прикладные нечеткие системы [Текст] / К. Асаи, Д. Ватада, С. Иваи. – М.: Мир, 1993. – 368 с.

6. Малышев, Н.Г. Нечеткие модели для экспертных систем в САПР [Текст] / Н.Г. Малышев, Л.С. Берштейн, А.В. Боженюк. – М.: Энергоатомиздат, 1991. – 136 с.

7. Страуструп, Бьярн. Язык программирования С++ [Текст] / Бьярн Страуструп. – К.: Диасофт, 1993. – 264 с.

8. Герасимов, А.С. Разработка и реализация алгоритма поиска вывода в расширении бесконечнозначной предикатной логики Лукашевича [Текст]: дис. на соиск. ... канд. физ.-мат. наук. / А.С. Герасимов; [Санкт-Петербургский государственный университет]. – СПб., 2007 – 194 с.

9. Сильнова, С.В. Исследование валидности правил в системе поддержки принятия решений при управлении сбытом [Текст] / С.В. Сильнова, Е.А. Пузырникова // Штучний інтелект. – 2009, № 4. – С. 302 – 310.

Надійшла до редколегії 7.11.2012

Рецензент: д-р техн..наук, проф. А.М. Сінотін, Харківський національний університет радіоелектроніки, Харків.

ЛОГИЧЕСКАЯ МОДЕЛЬ АДАПТИВНОГО ПРИНЯТИЯ РЕШЕНИЙ

А.М. Цымбал

В статье предлагается метод адаптивного принятия решений для интеллектуальных систем управления роботами. Использование адаптивного принятия решений позволит учитывать изменения в среде принятия решений. Рассматривается логическая модель, описывающая адаптивное принятие решений в роботизированных системах. В качестве примеров представления информации использован язык Пролог.

Ключевые слова: *адаптивное принятие решений, логическая модель, предикатная модель, робот.*

LOGICAL MODEL OF ADAPTIVE DECISION-MAKING

A.M. Tsymbal

Article proposes the method of adaptive decision-making for intellectual control systems of robots. The application of adaptive decision-making will allow to take in account the changes in the environment of decision-making. There is considered the logical model, which describes the adaptive decision-making for robotic systems/ As an example of information presentation, there is used Prolog language.

Keywords: *adaptive decision-making, logical model system, predicate model, robot.*