DOI: https://doi.org/10.33216/1998-7927-2021-266-2-26-34

УДК 004.89:620.91

# АВТОМАТИЗАЦІЯ КЕРУВАННЯ ЕНЕРГЕТИЧНИМИ СИСТЕМАМИ НА ОСНОВІ ПРОЦЕСУ ІНТЕРПРЕТАЦІЇ МЕТАПРАВИЛ БАЗИ ЗНАНЬ ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ

**Моркун В.С., Котов І.А., Сердюк О.Ю., Гапоненко І.А.**

# AUTOMATION OF CONTROL OVER POWER SYSTEMS ON THE BASIS OF INTERPRETING METARULES OF THE SMART SYSTEM KNOWLEDGE BASE

**Morkun V.S., Kotov I.A., Serdiuk O.Y., Haponenko I.A.**

*У статті розглянута проблема побудови інтелектуальної програмної системи, реактивної по відношенню до подій зовнішнього середовища, для автоматизації управління режимами енергосистем. Модель реактивної тригерної системи підтримки рішень інтерактивно пов'язана як з станами компонентів енергосистеми, так і з діями користувача-оператора. Обґрунтоване, що теоретична розробка та впровадження програмного комплексу тригерної системи підтримки прийняття рішень в середу автоматизованої системи диспетчерського керування енергосистеми є актуальною науково-технічною проблемою. Методи дослідження полягають в застосуванні автоматної моделі станів функціонування автоматизованої програмної системи підтримки рішень. Для схеми станів реалізована автоматна модель функціонування з конкретизацією семантики станів, транзакцій і тригерів. У статті приведена схема трансляції програми тригерних транзакцій метаправил в байт-код інтерпретатора. Приведені результати інтерпретації метаправил бази знань інтелектуальної системи.*

***Ключові слова**: граматика, онтологія, трансляція, транзакція, тригер, енергосистема, метаправило*

**Вступ**. В роботі розглядається актуальна проблема автоматизації управління режимами електричних мереж на основі використання програмних контролерів рівнів професійних знань, які не започатковано тригерами від об'єкта управління. Сучасні електричні мережі формують ряд особливих вимог як до самої інтелектуальної системі, так і до методології її проектування. Наведемо базові з таких вимог: одночасна робота з декларативними і процедурними формами представлення знань, однакова робота системи в режимі логічного висновку в базі знань (БЗ) і обробки бази даних (БД), забезпечення практичного апаратного інтерфейсу з компонентами енергооб'єктів та автоматизованої-ванній системи диспетчерського управління (АСДУ), реалізація призначеного для користувача візуального інтерфейсу з оперативним диспетчерським персоналом (ОДП), ефективна трансляція у внутрішні коди системи і інтерпретація керуючих метаправило, реалізація програмного механізму логічного висновку у вигляді транзакцій метаправило. Сформульовані вимоги обумовлюють необхідність нового ексклюзивного підходу до подання і візуалізації структурних і функціональних моделей програмного комплексу СППР. Модель повинна вирішувати такі основні завдання: забезпечення однакового уявлення алгоритмів функціонування СППР і алгоритмів трансляції та обробки бази знань на всіх рівнях репрезентації, дієвого виконання програми в залежності від параметрів (сигналів) станів зовнішнього оточення інтелектуальної системи.

**Постановка проблеми.** На основі розробленої вище структурної схеми декомпозиції СППР, включеної в контур управління промисловим об'єктом (ЕЕС), можна синтезувати структурно-логічну тригерну модель і визначити методи функціонування ядра системи підтримки рішень. БЛВ отримує завдання в формалізованому вигляді і вирішує її шляхом маніпуляцій по відношенню до концептів бази знань. При цьому, механізм обробки знань зазвичай визначається наперед і жорстко прив'язаний до форм подання знань в конкретній системі і предметної області. Зміни професійної сфери, структури або способу формалізації бази знань тягнуть за собою необхідність переробки БЛВ. Крім того, класичний підхід чітко розмежовує специфіку складових ядра: БЛВ – це програмний модуль (зазвичай машинного коду), що обробляє відомі йому структури подання знань в БЗ, а БЗ – це формалізовані певною мірою

правила логічного висновку в рамках конкретної предметної області.

Однак, в складно-структурованих, ієрархічно організованих професійних областях, таких, як протиаварійне диспетчерське управління режимами енергосистеми, класична схема компонування ядра СППР не дозволяє використовувати всі форми подання знань предметної області в єдиній інтелектуальній системі. Тому в контексті дослідження вирішується завдання одночасного використання різних форм знань, представлених у вигляді онтологій і еволюційно узагальнюючих один одного. Даний підхід не може бути реалізований в рамках класичної структури СППР.

**Аналіз останніх досліджень і публікацій.** Згідно з усталеною класичною архітектурою інтелектуальних систем під ядром СППР розуміють два її основних модуля – блок логічного висновку (БЛВ) і базу знань (БЗ) [1 – 4].

При побудові інтелектуальної системи однією з основних задач є алгоритмізація, програмування і реалізація логічного висновку.

Проведемо короткий аналіз існуючих функціональних моделей візуалізації алгоритмів (сценаріїв, поведінки), які найбільш адекватно реалізують поставлені в роботі завдання [5 – 8]. Кожен з формалізмів має сильні сторони і деякі обмеження, а також переважну область застосування. Узагальнюючи розглянуті вище підходи до моделювання та візуалізації функціонування складних програмних систем, слід сказати, що вони не цілком відповідають завданню проектування подієвої інтелектуальної системи, що використовує інкорпоровану базу знань, засновану на еволюції рівнів професійних онтологій і керовану метаправилами, що активізується зовнішнім середовищем.

**Мета статті.** В складно-структурованих, ієрархічно організованих професійних областях, таких, як протиаварійне диспетчерське управління режимами енергосистеми, класична схема компонування ядра СППР не дозволяє використовувати всі форми подання знань предметної області в єдиній інтелектуальній системі. Тому в контексті дослідження вирішується завдання одночасного використання різних форм знань, представлених у вигляді онтологій і еволюційно узагальнюючих один одного. Даний підхід не може бути реалізований в рамках класичної структури СППР.

З метою подолання зазначеної проблеми повинна бути розроблена нова структурно-логічна архітектура ядра СППР і визначені методи її функціонування.

**Результати досліджень.** Для функціонування інтерпретатора транзакцій розроблені операційні структури даних, які показані на рис. 1.

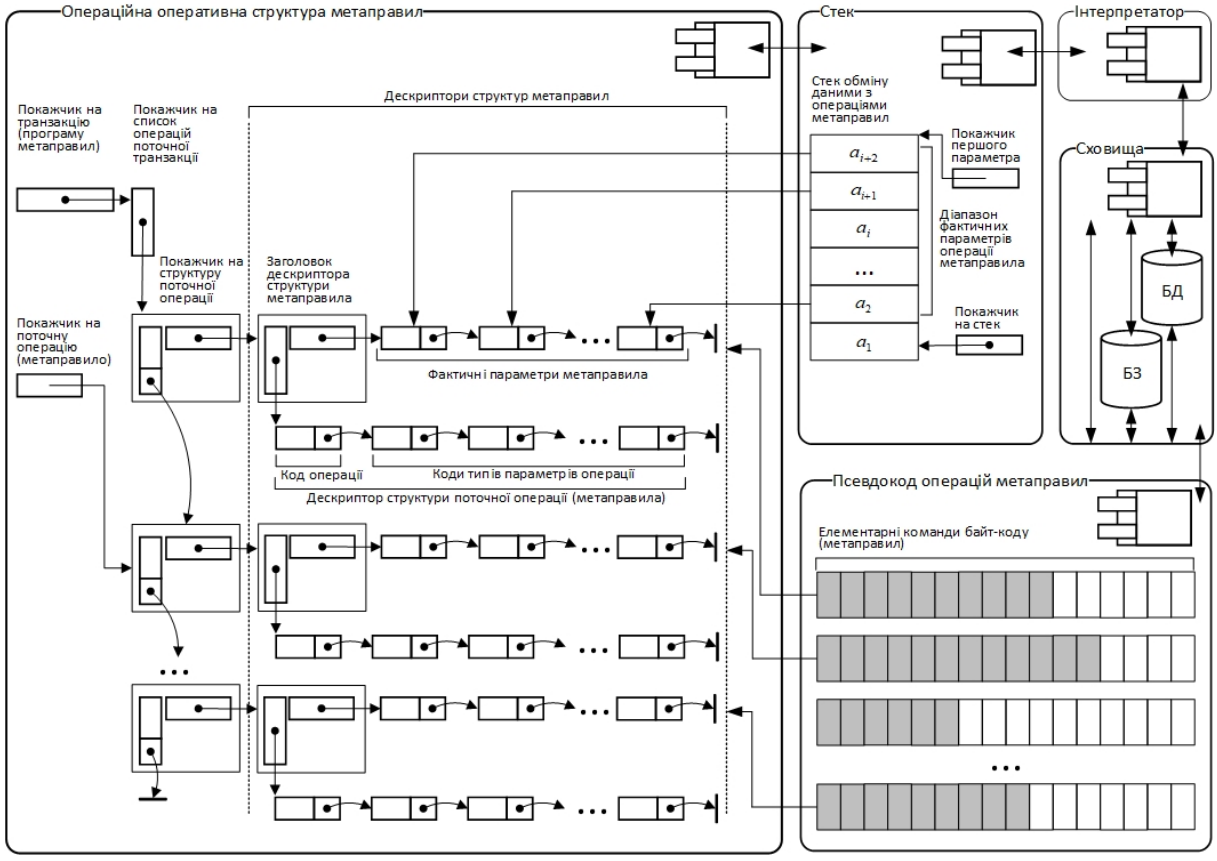


Рис. 1. Операційна структура управління даними обчислювального процесу інтерпретації метаправил

Основний набір компонентів операційних структур містить наступні блоки:

- блок оперативної операційної структури транзакції (програми) метаправил;
- блок байт-кодів операцій метаправил;
- блок стека передачі фактичних параметрів зовнішнього середовища;
- блок сховищ (БЗ і БД) з описувачем завдань і методів їх рішень;
- блок інтерпретатора.

Дамо коротку характеристику блоків.

Блок оперативної операційної структури транзакції (програми) метаправил розгортається в оперативній пам'яті динамічно, починаючи з покажчика на транзакцію. Далі розгортається список покажчиків на структури окремих операцій – метаправил. Структура включає в себе заголовок дескриптора метаправила, що містить два покажчика – покажчик на формат метаправила і покажчик на список фактичних параметрів. Фактичні параметри завантажуються в список з блоку стека параметрів.

Блок байт-кодів операцій метаправил, які являють собою оттрансльовані операції транзакцій по управлінню інформацією в БЗ і завантажуються з носія в пам'ять: вибірка концептів, зіставлення концептів, реструктуризація БЗ, логічний висновок, маркування сигнальних графів рівнів онтологій та інші. Крім робочих операцій там же розташовуються транзакції обробки позаштатних ситуацій: наявність суперечливості в БЗ, неможливість обробки поточної структури рівня, зупинка роботи користувачем і т.п. Обробка операцій проводиться за принципом виклику переривань. Блок стека передачі фактичних параметрів зовнішнього середовища здійснює параметричну взаємодію сховищ даних або джерел даних зовнішнього середовища і оброблюваних структур метаправил. Дані через стек заміщають формальні параметри метаправил відповідно до їх кількості, порядку і типів.

Блок сховищ (БЗ і БД) містить формалізовану і оттрансльовану базу знань з поділом на онтологічні рівні і базу даних з параметрами завдання, об'єкта управління, професійного середовища, довідниками обладнання і т.п. Дані в сховище надходять при заповненні оператором і оперативно – від об'єктів зовнішнього середовища в процесі функціонування системи. Інформація зі сховищ надходить в блок байт-кодів операцій метаправил для виконання операцій і в стек для передачі в якості фактичних параметрів. Управління потоками переданих кодів знань і даних здійснюється блоком інтерпретатора.

Блок інтерпретатора являє собою програмний процесор, що не залежить від операційної платформи обчислювальної системи і виконує потоки елементарних команд байт-коду метаправил. Виконання команд метаправил супроводжується двонаправленим обміном інформацією між компонентами інших блоків ядра СППР.

Розроблена операційна структура ядра СППР реалізує новий принцип обробки БЗ, який дозволяє відмовитися від блоку логічного висновку і замінити його програмним процесором (інтерпретатором), якій не залежить від форм знань, професійного середовища і операційної платформи. Інтерпретатор виконує потік байт-коду метаправил і здійснює операції з управління знаннями і логічного висновку. При цьому, метаправила також є частиною БЗ і допускають оперативну модифікацію. Параметрами команд байт-коду можуть служити дані тригерів, які отримують сигнали зовнішнього середовища.

Таким чином, метаправила, представлені в предикативний формі, можуть розглядатися як аналоги команд символічної асемблерної мови. Потік байт-коду, отриманий після трансляції метаправил, являє собою виконуваний інтерпретатором низькорівневий байт-код. Набір команд байт-коду, ідентифікованих і виконуваних інтерпретатором, є обмеженим і цілком визначеним. На його основі можна розробляти високорівневі мови логічного висновку і управління базами знань.

В роботі запропонована нова трикомпонентна схема реалізації ядра СППР, в якої БЛВ і БЗ перекриваються, і логіка обробки знань і даних обумовлюється вмістом знань – метаправилами. Метаправила одночасно є частиною БЗ і БЛВ і тому можуть розглядатися в якості програми обробки знань. Це дозволяє звільнити БЛВ від бізнес-логіки управління БЗ, і він може грати роль програмного процесора, незалежного від форм подання знань і специфіки професійної сфери.

При реалізації БЛВ як програмного процесора виникає проблема ефективної інтерпретації пакетів (транзакцій) метаправил. Для вирішення цієї проблеми актуалізується завдання розробки моделі ефективного внутрішнього подання метаправил. Внутрішнє подання має забезпечувати високу швидкість і незалежність роботи програмного процесора від семантики знань. Остання вимога зумовлює необхідність подання метазнань у вигляді ізоморфного низькорівневого байт-коду, що забезпечує однозначну інтерпретацію. Цей підхід, в свою чергу, вимагає розробки процедури трансляції пакетів метаправил в низькорівневий байт-код.

Запропонований підхід дозволяє розглядати потік низькорівневого байт-коду в якості програми, для якої може бути синтезована програма управління БЗ на мові високого рівня. Спираючись на розроблену вище тригерну модель мереж транзакцій метаправил, з'являється можливість розробки високорівневих подієво-керованих програм обробки даних і знань в СППР.

Розробимо формальну модель граматики програми транзакцій метаправил. Для подання вихідної форми метаправил в роботі запропоновано єдиний предикатний синтаксис. Граматика програми спирається на ідеологію предикатного програмування, де кожна транзакція є відокремленою предикатною програмою [9,10,11]. У свою чергу, предикатна транзакція, що складається з окремих метаправил, реалізує пакети функцій обробки знань. Тому при роз-

робці моделі граматики мови опису програми тригерних транзакцій використаний теоретичний базис функціонального програмування і денотаційної семантики, яким присвячено багато фундаментальних робіт [12 – 16].

Визначимо граматику таким чином:

$$G_T = < \Sigma_T, N_T, P_T, S_T >, \quad (1)$$

де $G_T$ – граматика тригерної програми транзакцій;

$\Sigma_T$ – термінальний алфавіт тригерної програми транзакцій;

$N_T$ – нетермінальний алфавіт моделі тригерної програми транзакцій;

$P_T$ – продукції граматики:

$$\exists a \exists b (a, b \in P_T), a \in (\Sigma_T \cup N_T)^+ \wedge b \in (\Sigma_T \cup N_T)^* : a \to b;$$

$$\Sigma_T \bigcap N_T = \varnothing;$$

$$P_T \subset (\Sigma_T \bigcup \Sigma_T)^+ \times (N_T \bigcup \Sigma_T)^*);$$

$S_T$ – стартовий нетермінальний символ граматики $G$.

Введемо позначення.

Множина нетерміналів = {програма (Pr), заголовок (H), декларації (Dcl), тіло (B), ім'я (Nm), оголошення_тригерів (DTrigg), оголошення_транзакцій (DTrans), перелік_імен (LNm), перелік_блоків_транзакцій (LBTrans), транзакція (Trans), ім'я_транзакції (NmTrans), ім'я_тригера (NmTrigg), перелік_операцій (LO), перехід (Jmp), операція (O), ім'я_операції (ONm), перелік_параметрів (LP), перелік_концептів (LC), рядок (S), число (D), ціле_число (Int), ідентифікатор (Id), знак (Sign), літера (Ch), цифра (Dig)}.

Множина терміналів = {program, trigger, trans, next, (, ), ;, {, }, a|b|c|…|z|A|B|C|…|Z|a|б|в|…|я|A|Б|B|…|Я, +|-|, 0|1|2|…|9, ., ,}. P = програма.

Представимо укрупнену граматику мови [17 – 19]:

P = {
1. = Pr → HDclB.⊥ ,
2. H → program(Nm) ,
3. Dcl → DTrigg; ,
4. Dcl → DTrans; ,
5. DTrigg → trigger(LNm); ,
6. DTrans → trans(LNm); ,
7. LNm → Nm {, Nm} ,
8. B → {LBTrans} ,
9. LBTrans → Trans {, Trans} ,
10. Trans → NmTrans (NmTrigg) {LO [Jmp]}; ,
11. NmTrans → Nm ,
12. NmTrigg → Nm ,
13. LO → O; {O;} ,
14. Jmp → next(NmTrans); ,
15. O → ONm (LP); ,
16. ONm → Nm ,
17. LP → NmTrans, LC ,
18. LC → Nm, | S, | D,{Nm, | S, | D,}
19. Nm → Id ,
20. Id → Ch {Ch | Dig | _} ,
21. S → Ch | Dig |_{Ch | Dig | _}
22. Ch → a|b|c|…|z|A|B|C|…|Z|a|б|в|…|я|A|Б|B|…|Я ,
23. D → Sign Int.Int ,
24. Sign → +|-| ,
25. Int → Dig {Dig},
26. Dig → 0|1|2|…|9
}

На основі розробленої синтаксичної моделі можна записати початковий текст програми. Нижче наведено лістинг прикладу початкового коду програми тригерних транзакцій операційних метаправил управління онтологіями БЗ.

```
// заголовок програми
program(prog1)
// оголошення тригерів
trigger(t1);
// оголошення тригерних блоків транзакцій
trans(t_start, trans1, trans2);
{
  // початок програми
 // ім'я триггерного блоку t1 -тригер
 t_start(t1)
  {
// початок пакета транзакції
// операції метаправил
.....
// одна операція транзакції і
//примусовий перехід до транзакції – trans2
ADDP(trans1, station04, "п/ст–Южная 110кВ", 0.2, 0.95, 0);
.....
// операції метаправил
// плановий перехід до транзакції – trans3
next(trans3);
  };                  // кінець пакету транзакції
}.                    // кінець програми
```

Для забезпечення ефективного виконання транзакцій метаправил і взаємодії із зовнішнім середовищем сформулюємо завдання трансляції початкового тексту програми у внутрішнє подання ядра СППР. Результатом трансляції повинен бути потік коду, що однозначно інтерпретується, виконання якого реалізує управління онтологіями БЗ і логічний висновок.

Розробка транслятора спирається на типові фази трансляції програми – препроцесінг, лексичний, синтаксичний і семантичний аналізи, оптимізацію, генерацію коду і компоновку. Вибір фаз трансляції та їх функціонал залежать від конкретної спрямованості транслятора [18]. З урахуванням специфіки інтерпретації пакетів метаправил виділимо ядро завдання трансляції та введемо спрощують обмеження. Приймемо в якості основних етапів трансляції лексичний, синтаксичний і семантичний аналізи і генерацію байт-коду для інтерпретатора. Тоді загальна схема трансляції тригерної транзакції метаправил в байт-код інтерпретатора може бути проілюстрована на рис. 2.

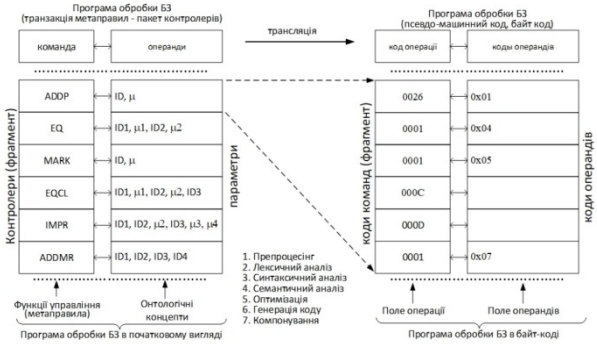


Рис. 2. Узагальнена схема трансляції програми тригерних транзакцій метаправил в байт-код інтерпретатора

Деталізуємо схему, з огляду на обумовлену специфіку фаз трансляції. На вхід аналітичної групи етапів схеми подається програма пакетів транзакцій метаправил в початковому коді [20 – 22]. Кожен етап аналітичної групи трансляції супроводжується розбором і обробкою помилок. Група етапів трансляції, яка синтезує, відповідає за подання операцій транзакцій у вигляді базових функцій на сигнальних графах БЗ, генерації вихідного асемблерного коду і подальше кодування його у внутрішній байт-код. На всіх етапах трансляції проводиться взаємодія з таблицею ідентифікаторів тригерів, транзакцій і операцій. Схема трансляції породжує двійковий код, який однозначно може бути оброблений інтерпретатором ядра СППР.

Розробимо функціональний зміст основних етапів трансляції пакетів транзакцій метаправил. Метою лексичного аналізу (ЛА) є побудова таблиць токенов лексем, що використовуються в програмі. Лексичний аналіз програми тригерних транзакцій заснований на граматиці мови, наведеної вище. Лексеми вхідної мови утворюють такі групи – ключове слово, обмежувач, спеціальний символ, ідентифікатор, константа.

Індекс лексеми повинен забезпечувати ефективний доступ до лексеми в таблиці токенов, тому, в загальному випадку, він реалізується шляхом використання хеш-адресації. В якості спрощеної ілюстрації для розглянутого прикладу використані порядкові індекси лексем в таблиці токенов. Лексеми вхідного потоку не порожні, тому для опису тексту програми досить автоматної граматики без $\varepsilon$-правил [23]. Приймемо, що межами лексем служать пробіли, знаки переводу рядка і повернення каретки, круглі дужки, крапка з комою. Сканер для лексичного аналізу реалізується моделлю кінцевого автомата. Задамо скануючий автомат лексичного кінцевого перетворювача його укрупненої діаграмою станів.

В результаті роботи лексичного сканера і кінцевого перетворювача можуть бути отримані набори токенов.

На основі отриманих результатів ЛА побудуємо синтаксичний аналізатор (СА) для розбору ланцюжка лексем. Існує однозначна відповідність між токенами лексем і символами граматики. За основу СА приймемо модель автомата – розширеного МП-аналізатора. Були оцінені можливості застосування стратегій розбору для розглянутого формалізму програми. Універсальні алгоритми розбору (табличні розпізнавачі Кока-Янгера-Касамі і Ерлі) є більш ефективними, проте зустрічають ряд труднощів у практичній реалізації [20]. Тому, з огляду на відносно просту структуру і відсутність ліворекурсивності, реалізуємо спадний СА з лівим розбором і підбором альтернатив [17, 22].

Застосуємо модель синтаксичного МП-аналізатора типу зі схемою зміни конфігурацій до початкового тексту програми тригерних транзакцій. Для простоти ілюстрації синтаксичного розбору візьмемо одну операцію лістингу −

ADDP(trans1, station04, “п/ст – Південна 110 кВ”, 0.2, 0.95, 0);.

Після лексичного аналізу та побудови таблиці токенов розглянута операція набуде вигляду

<idID,0012><idLP,0005><idID,0010><idCom,000A> <idID,0013> <idCom,000A><idConst,0014> <idCom,000A> <idConst,0015><idCom,000A><idConst,0016><idCom,000A> <idConst,0017><idRP,0006><idSem,0009>

Цієї операції буде відповідати фрагмент загального дерева розбору програми, показаний на рис. 3.

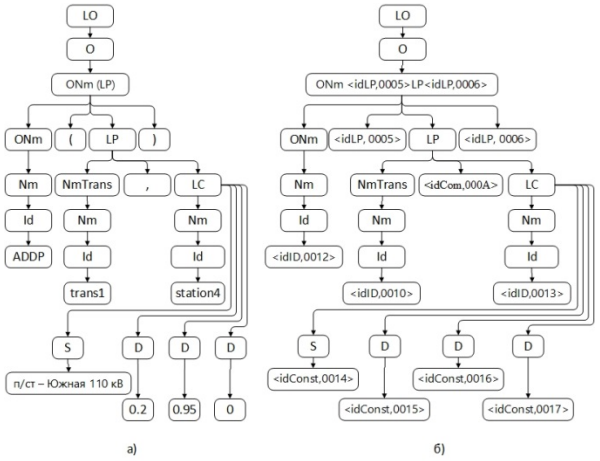


Рис. 3. Дерево синтаксичного аналізу однієї операції програми: а – в початковому вигляді, б – у вигляді токенов лексем після лексичного аналізу

Результати синтаксичного розбору показали, що МП-аналізатор, на базі розробленої граматики, допускає вхідний ланцюжок і розпізнає його вірно. Отриманий результат можна поширити на всю структуру програми.

Синтаксичний аналіз повного лістингу програми проводиться аналогічним чином з використанням таблиці токенов, розробленої раніше.

Реалізація семантичного аналізу може бути здійснена на множині формальних моделей – атрибутних, імперативних (операційних), аплікативних, аксіоматичних, моделей специфікацій [16]. З огляду на специфіку визначення граматики і синтаксису програми транзакцій метаправил, використовуємо атрибутну модель з семантичними правилами. Атрибутна модель дозволяє асоціювати з вузлами синтаксичного дерева програми семантичні правила.

У семантичних правилах синтезований атрибут *val* відповідає значенню, що повідомляється нетерміналу в вузлі синтаксичного дерева, атрибут *str* відповідає фактичному значенню рядка, атрибут *code* відповідає фактичному значенню числа, синтезований атрибут *lexval* відповідає значенню, що повертається лексичним аналізатором.

Для трансляції програми тригерних транзакцій метаправил у внутрішній байт-код ядра СППР приймемо в якості проміжного коду подання програми структуру у вигляді тріад

<операція><операнд_1><операнд_2>.

При цьому допускаються випадки одного операнда або відсутності операндів

<операція><операнд_1>,

<операція>.

Дана форма проміжного коду обрана тому, що являє собою модель низькорівневого синтаксису мови асемблера, яка може відображатися мнемонічне і піддається візуальному сприйняттю, а також є максимально зручною для подальшої трансляції у внутрішній байт-код інтерпретатора ядра СППР. Кожна операція проміжного коду є елементарною операцією по відношенню до концептів бази знань, сукупність яких утворює сигнальний граф. Робота сигнального графа задається операціями його перемаркировки. При обході синтаксичного дерева заповнюються таблиці лексем програми. Генерація проміжного коду проводиться за схемою, показаною на рис. 4.

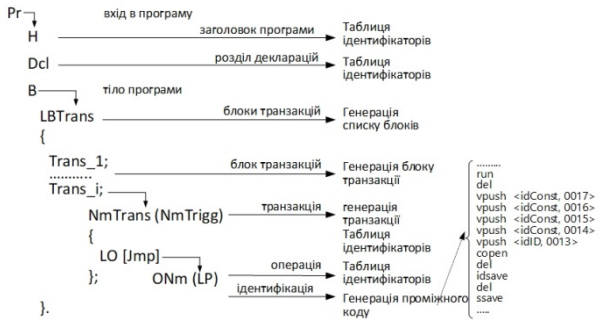


Рис. 4. Схема генерації проміжного коду

Кожна операція транзакції розгортається в стандартну, заздалегідь визначену програму проміжного коду з фактичними параметрами токенов таблиці лексем.

Програма реалізує обробку функцій управління концептами БЗ через операції транзакцій в предикативний формі. Тому в початковому тексті відсутні в явному вигляді арифметичні операції і операції передачі управління. Інструкції проміжного коду виконуються послідовно. Перехід до іншого блоку транзакцій означає завантаження і запуск окремої послідовності коду. Виходячи з цих положень, розроблений каркасний набір команд проміжного коду інтерпретатора. Цей набір є базовим і в подальшому може бути розширено.

У прийнятій реалізації програми відсутня необхідність в додаткових регістрах для операцій і передачі даних. Один службовий регістр використовується для покажчика на поточну виконувану інструкцію.

Як наслідок, за основу базової архітектури роботи інтерпретатора транзакцій ядра СППР прийнята концепція стекової машини. Стек є основною робочою пам'яттю для передачі параметрів в обробку і повернення результатів. Фактичні параметри для транзакцій метаправил передаються в стек відповідно до порядку проходження формальних параметрів в оголошенні. Результатом виклику транзакцій, відповідно до визначення операцій, є значення TRUE або FALSE.

На основі розробленого набору команд реалізується етап трансляції в байт-код ядра СППР. На цьому етапі виконується практично прямий переклад елементарних команд проміжного коду в байт-код внутрішнього подання програми для інтерпретатора.

Розглянемо приклад трансляції програми з однією операцією транзакції, лістинг якої був наведений вище,

ADDP(trans1, station04, “п/ст – Южная 110 кВ”, 0.2, 0.95, 0);.

Результат побудови кодів лексем констант, пов'язаних з токенами лексичного сканера, наведено в таблиці 1.

Таблиця 1

**Таблиця кодів лексем (імен) програми**

| Код | Лексема | Токен |
|---|---|---|
| 0x0001 | t_start | <idID,000F> |
| 0x0002 | 0.95 | <idConst,0016> |
| 0x0003 | 0.2 | <idConst,0015> |
| 0x0004 | п/ст – Південна 110 кВ | <idConst,0014> |
| 0x0005 | station04 | <idID,0013> |
| 0x0006 | d:\base\basefile.dat | val =d:\base\basefile.dat |
| 0x0007 | trans2 | <idID, 0011> |

Загальні результати трансляції програми наведені в таблиці 2. Таблиця містить вихідний (проміжний) код програми, байт-код для інтерпретації в ядрі СППР, стан стека в ході виконання програми, короткий опис дій кожної команди. Символом «#» позначена директива інтерпретатора для реалізації його взаємодії із зовнішнім середовищем.

Програма виконується послідовно – покомандно. Інтерпретатор ставить у відповідність кожній команді мікропрограму обробки метаправил. Процес ідентифікує чергову команду і запускає мікропрограму для її виконання. Директиви керують режимом роботи самого інтерпретатора, а також ходом виконання програми. Для подієвого зв'язку із зовнішнім середовищем і реалізації тригерів інтерпретатор переходить в режим очікування. При реєстрації події інтерпретатор ідентифікує подію і завантажує програму транзакції, назначену тригером даної події.

Таблиця 2

**Результати трансляції програми тригерних транзакцій**

| Проміжний код | Байт-код | Стек | Зміст команди |
|---|---|---|---|
| #wait t_start | 0026 0x01 | 0x01 | очікувати команду запуску |
| vpush 1 | 0001 1 | 0x01 | задати прапор для виконання |
| run | 0024 | 0x01 | виконати критичний блок |
| del | 0002 | - | виштовхнути з стека |
| vpush 0 | 0001 0x00 | 0x00 | в стек 0 - активність концепту |
| vpush 0.95 | 0001 0x02 | 0x00, 0x02 | в стек 0.95 поріг активації |
| vpush 0.2 | 0001 0x03 | 0x00, 0x02, 0x03 | в стек 0.2 - потенціал вузла |
| vpush “п/ст–Південна 110кВ” | 0001 0x04 | 0x00, 0x02, 0x03,0x04 | в стек "п / ст - Південна 110 кВ" |
| vpush “station04” | 0001 0x05 | 0x00, 0x02, 0x03,0x04,0x05 | в стек "station04" |
| vpush “d:\base\basefile.xml” | 0001 0x06 | 0x00,0x02,0x03,0x04,0x05,0x06 | в стек "d:\base\basefile.xml" |
| copen | 0003 | 0x00,0x02,0x03,0x04,0x05,0x06 | відкрити файл концептів |
| del | 0002 | 0x00,0x02,0x03,0x04,0x05 | виштовхнути з стека |
| idsave | 000C | 0x00,0x02,0x03,0x04,0x05 | зберегти ідентифікатор концепту |
| del | 0002 | 0x00,0x02,0x03,0x04 | виштовхнути з стека |
| ssave | 000D | 0x00,0x02,0x03,0x04 | зберегти зміст концепту |
| del | 0002 | 0x00,0x02,0x03 | виштовхнути з стека |
| usave | 000E | 0x00,0x02,0x03 | зберегти потенціал концепту |
| del | 0002 | 0x00,0x02 | виштовхнути з стека |
| tsave | 000F | 0x00,0x02 | зберегти поріг активації концепту |
| del | 0002 | 0x00 | виштовхнути з стека |
| asave | 0010 | 0x00 | зберегти прапор активності концепту |
| del | 0002 | - | виштовхнути з стека |
| vpush “d:\base\basefile.xml” | 0001 0x06 | 0x06 | в стек "d:\base\basefile.xml" |
| cclose | 0004 | 0x06 | закрити файл концептів |
| del | 0002 | - | виштовхнути з стека |
| vpush trans2 | 0001 0x07 | 0x07 | в стек trans2 |
| vpush 0 | 0001 0x00 | 0x07,0x00 | в стек 0 |
| cond | 0025 | 0x07 | перехід до транзакції trans2 |
| ret | 0022 | 0x07 | вийти з критичного блоку |
| end | 0023 | 0x07 | зупинити роботу програми |

Фактичні дані (параметри) концептів БЗ завантажуються з файлів. Формат зберігання інформації в файлах принципового значення не має і залежить від багатьох причин, в тому числі, – від мови реалізації системи, корпоративних традицій розробки, вимог ефективності, засобів автоматизації документування і т.п. Крім того, дані можуть бути занесені користувачем. Каналом передачі даних між командами програми і між програмою і БЗ є стек.

**Висновки.** Розроблено функціональну схему трансляції програми тригерних транзакцій метаправил в байт-код інтерпретатора. Розроблені компоненти моделі аналітичного розбору програми транзакцій – лексичний сканер, синтаксичний МП-аналізатор, семантичні правила. Побудована комплексна таблиця токенов лексем програми транзакцій. Сформовано синтаксичне дерево і проведено повний розбір операції транзакції в програмі. Результати синтаксичного розбору показали, що МП-аналізатор, використовуючи розроблену граматику, допускає вхідний ланцюжок і розпізнає його вірно. Побудована атрибутна модель семантичного аналізу програми.

Реалізована процедура трансляції програми тригерних транзакцій метаправил в байт-код ядра СППР. На основі моделі синтаксично керованого перекладу (СКП) – СК-трансляції – розроблена схема генерації низькорівневого коду програми. Розроблено каркасний набір елементарних команд проміжного коду інтерпретатора. Цей набір є базовим і в подальшому може бути розширено. На основі розробленого набору елементарних команд реалізована трансляція в байт-код ядра СППР. Обґрунтовані структура інтерпретатора і правила виконання байт-коду програми. Сформовані функції інтерпретатора з управлінню стеком і подієвої взаємодії із зовнішнім середовищем.

**Література**


1. Джексон П. Введение в экспертные системы // Вильямс – 2001 – 624 с.
2. Бохуа Н.К., Геловани В.А., Ковригин О.В. Экспертные системы: опыт проектирования – М.: МНИИПУ – 1990 – 348 с.
3. Таунсенд К., Фохт Д. Проектирование и программная реализация экспертных систем на персональных ЭВМ / Пер. с англ. – М.: Финансы и статистика – 1990 – 320 с.
4. Левин Р., Дранг Д., Эделсон Б. Практическое введение в технологию искусственного интеллекта и экспертных систем с иллюстрациями на Бейсике – М.: Финансы и статистика – 1990 – 239 с.
5. Пышкин Е.В. Структуры данных и алгоритмы: реализация на C/C++ / Е.В. Пышкин − СПб.: ФТК СПБГПУ – 2009 − 200 с.
6. Карпов Ю.Г. Теория автоматов. Учебник для вузов – СПб.: Питер, 2003.- 208 с.

7. Кнут Д. Искусство программирования для ЭВМ. Т.1. Основные алгоритмы. − М.: Мир – 1976 − 736 с.
8. Wirth N. Algorithms + Data Structures = Programs. Prentice-Hall, Inc. Englewood Cliffs, N.J. 1976.
9. Шелехов В.И. Введение в предикатное программирование / В.И. Шелехов − Новосибирск − 2002 – 82 с.
10. Шелехов В.И. Семантика языка предикатного программирования // ЗОНТ-15 – Новосибирск – 2015 − 13с.
11. Shelekhov V. The language of calculus of computable predicates as a minimal kernel for functional languages − BULLETIN of the Novosibirsk Computing Center. Series: Computer Science., IIS Special Issue – 2009 − 29(2009) − P.107−117.
12. Филд А., Харрисон П. Функциональное программирование: Пер. с англ. − М.: Мир − 1993 − 637 с.
13. Stoy J.E. Denotational semantics: The Scott-Strachy Approach to Programming language theory. — MIT Press, Cambridge, M.A. – 1977 – 414 p.
14. Allison L. A Practical Introduction to Denotational Semantics − Cambridge University Press − 2012 – 148 p.
15. Орлов С.А. Теория и практика языков программирования: Учебник для вузов. Стандарт 3-го поколения. — СПб.: Питер, 2013. — 688 с.
16. Пратт Т., Зелковиц М. Языки программирования: разработка и реализация / Под общ. ред. А. Матросова – СПб.: Питер – 2002 – 688 с.
17. Ахо А., Ульман Дж. Теория синтаксического анализа, перевода и компиляции. / Под ред. В.М. Курочкина – Пер. с англ. В.Н. Агафонова − Т.1: Синтаксический анализ − М.: Мир – 1978 – 612 с.
18. Ахо А., Ульман Дж. Теория синтаксического анализа, перевода и компиляции. / Под ред. В.М. Курочкина – Пер. с англ. А.Н. Бирюкова, В.А. Серебрякова − Т.2: Компиляция − М.: Мир – 1978 – 487 с.
19. Опалева Э.А., Самойленко В.П. Языки программирования и методы трансляции − СПб.: БХВ-Петербург – 2005 − 480 с.
20. Ахо А.В., Лам М.С., Сети Р., Ульман Дж. Компиляторы: принципы, технологии и инструментарий, 2-е изд. : Пер. с англ. − М.: ООО “И.Д. Вильямс” − 2008 − 1184 с.
21. Карпов В. Э. Классическая теория компиляторов: Учеб. пособие / В.Э. Карпов − М.: Моск. гос. ин-т электроники и математики − 2002 − 78 с.
22. Молдованова О.В. Языки программирования и методы трансляции: Учебное пособие / О.В. Молдованова – Новосибирск: СибГУТИ − 2012 − 134с.
23. Волкова И.А., Вылиток А.А., Руденко Т.В. Формальные грамматики и языки. Элементы теории трансляции − М.: Издательский отдел факультета ВМиК МГУ им. М.В. Ломоносова (лицензия ИД № 05899 от 24.09.2001), 2009 − 115 с.

**References**

1. Dzhekson P. Vvedeniye v ekspertnyye sistemy // Viliams – 2001 – 624 c.
2. Bokhua N.K.. Gelovani V.A.. Kovrigin O.V. Ekspertnyye sistemy: opyt proyektirovaniya – M.: MNIIPU –1990 – 348 c.
3. Taunsend K.. Fokht D. Proyektirovaniye i programmnaya realizatsiya ekspertnykh sistem na personalnykh EVM / Per. s angl. – M.: Finansy i statistika – 1990 – 320 c.
4. Levin R.. Drang D.. Edelson B. Prakticheskoye vvedeniye v tekhnologiyu iskusstvennogo intellekta i ekspertnykh sistem s illyustratsiyami na Beysike – M.: Finansy i statistika – 1990 – 239 c.
5. Pyshkin E.V. Struktury dannykh i algoritmy: reali-zatsiya na C/C++ / E.V. Pyshkin − SPb.: FTK SPbGPU – 2009 − 200 c.
6. Karpov Yu.G. Teoriya avtomatov. Uchebnik dlya vuzov – SPb.: Piter, 2003.- 208 c.
7. Knut D. Iskusstvo programmirovaniya dlya EVM. T.1. Osnovnyye algoritmy. – M.: Mir – 1976 − 736 c.
8. Wirth N. Algorithms + Data Structures = Programs. Prentice-Hall, Inc. Englewood Cliffs, N.J. 1976.
9. Shelekhov V.I. Vvedeniye v predikatnoye programmirovaniye / V.I. Shelekhov - Novosibirsk − 2002 – 82 c.
10. Shelekhov V.I. Semantika yazyka predikatnogo programmirovaniya // ZONT-15 – Novosibirsk – 2015 − 13c.
11. Shelekhov V. The language of calculus of computable predicates as a minimal kernel for functional languages − BULLETIN of the Novosibirsk Computing Center. Series: Computer Science., IIS Special Issue – 2009 − 29(2009) − P.107−117.
12. Fild A.. Kharrison P. Funktsionalnoye programmirovaniye: Per. s angl. – M.: Mir − 1993 − 637 c.
13. Stoy J.E. Denotational semantics: The Scott-Strachy Approach to Programming language theory. — MIT Press, Cambridge, M.A. – 1977 – 414 p.
14. Allison L. A Practical Introduction to Denotational Semantics − Cambridge University Press − 2012 – 148 p.
15. Orlov S.A. Teoriya i praktika yazykov programmirovaniya: Uchebnik dlya vuzov. Standart 3-go pokoleniya. – SPb.: Piter, 2013. – 688 c.
16. Pratt T. Zelkovits M. Yazyki programmirovaniya: razrabotka i realizatsiya / Pod obshch. red. A. Matrosova – SPb.: Piter – 2002 – 688 c.
17. Akho A.. Ulman Dzh. Teoriya sintaksicheskogo analiza. perevoda i kompilyatsii. / Pod red. V.M. Kurochkina – Per. s angl. V.N. Agafonova – T.1: Sintaksicheskiy analiz – M.: Mir – 1978 – 612 c.
18. Akho A.. Ulman Dzh. Teoriya sintaksicheskogo analiza. perevoda i kompilyatsii. / Pod red. V.M. Kurochkina – Per. s angl. A.N. Biryukova. V.A. Serebryakova – T.2: Kompilyatsiya – M.: Mir – 1978 – 487 c.
19. Opaleva E.A. Samoylenko V.P. Yazyki programmirovaniya i metody translyatsii – SPb.: BKhV –Peterburg – 2005 − 480 c.
20. Akho A.V. Lam M.S. Seti R. Ulman Dzh. Kompilyatory: printsipy. tekhnologii i instrumentariy. 2-e izd. : Per. s angl. – M.: OOO “I.D. Viliams” − 2008 − 1184 c.
21. Karpov V.E. Klassicheskaya teoriya kompilyatorov: Ucheb. posobiye / V.E. Karpov – M.: Mosk. gos. in-t elektroniki i matematiki − 2002 − 78 c.
22. Moldovanova O.V. Yazyki programmirovaniya i metody translyatsii: Uchebnoye posobiye / O.V. Moldovanova – Novosibirsk: SibGUTI − 2012 − 134c.
23. Volkova I.A.. Vylitok A.A.. Rudenko T.V. Formalnyye grammatiki i yazyki. Elementy teorii translyatsii – M.: Izdatelskiy otdel fakulteta VMiK MGU im. M.V. Lomonosova (litsenziya ID № 05899 ot 24.09.2001), 2009 − 115 c.

**Morkun V.S., Kotov I.A., Serdiuk O.Y., Haponenko I.A. Automation of control over power systems on the basis of interpreting metarules of the smart system knowledge base**

*The research deals with the issue of building a smart software system which is reactive to events of the external environment to automatize the dispatch control over power system modes. The model of the reactive trigger system of a decision-support system is interactively associated with both states of the power system and the operator's actions. It is substantiated that theoretical elaboration and implementation of the software complex for the trigger decision support system into the environment of the automated dispatch control system of the power control are topical scientific and technical problems. The smart system knowledgebase is built to provide decision support for dispatch control over power system modes under standard and emergency conditions. Knowledgebase based on a subset of the linguistic corpus of dispatch instructions for extinguishing emergencies and accidents in the power system. The research methods involve combining an automatic model of states and a trigger model of a software decision-support system, application of the automatic model of functioning states of the automatized smart decision support system. The suggested model of algorithm visualization presented as a trigger network of system states ensures interaction with the external environment. New interpretation of components of the trigger model is introduced. There are developed components of the analyzing model for the metarules transaction – a lexical scanner, a syntactical analyzer, and semantic rules. The operational structure of the decision support system kernel implements an innovative principle of processing the knowledgebase that enables refusing the logic input block and replacing it with the software processor (interpreter) independent of knowledge forms, the professional environment and the operating platform. The model can be applied to solving the following problems: providing identical representation of algorithms of the DSS functioning and algorithms of knowledgebase translation and processing at all representation levels, actual fulfillment of the program depending on parameters (signals) of conditions of the external surrounding of the smart system.*

***Keywords:*** *grammar, ontology, translation, transaction, trigger, power system, meta-rule*

**Моркун Володимир Станіславович** – доктор технічних наук, професор, проректор з наукової роботи, Криворізький національний університет, вул. Віталія Матусевича, 11, м. Кривий Ріг, Україна, 50027, E-mail: morkunv@gmail.com

**Котов Ігор Анатолійович** – кандидат технічних наук, доцент, кафедра моделювання та програмного забезпечення, Криворізький національний університет, вул. Віталія Матусевича. 11, м. Кривий Ріг, Україна, 50027, E-mail: rioexito@gmail.com

**Сердюк Олександра Юріївна** – асистент, кафедра автоматизації, комп'ютерних наук і технологій, Криворізький національний університет, вул. Віталія Матусевича. 11, м. Кривий Ріг, Україна, 50027, E-mail: o.serdiuk@i.ua

**Гапоненко Ірина Анатоліївна** – кандидат технічних наук, старший науковий співробітник відділу науки і досліджень, Криворізький національний університет, вул. Віталія Матусевича. 11, м. Кривий Ріг, Україна, 50027, E-mail: irinagaponenko44@gmail.com

Стаття подана 22.01.2021