Інженерія знань

6 курс, осінь 2021

- Доц. Баклан І.В.
- Email: iaa@ukr.net
- Web: baklaniv.at.ua

Лекція 5

Система подання знань у вигляді фреймів MFRL

Представлення знань у вигляді фреймів

Стислі відомості про ФРЛ/ЛІСП

Мова MFRL / PC (Frame Representation Language with Matching for PC) розроблений на базі мови ФРЛ (FRL - Frame Representation Language, Roberts & Goldstein, MIT 1977) в MEI на кафедрі прикладної математики.

Мова ФРЛ призначений для обробки фреймів - структур даних спеціального виду, які можуть бути використані як для декларативного, так і для процедурного представлення знань. Зауважимо, що ФРЛ не є самостійною мовою програмування, а суть розширення деякого мови обробки складноструктурованих даних, як правило - ЛИСП, що і мається на увазі в подальшому викладі. Описувана версія мови ФРЛ відповідає версії 3.0 системи MFRL / PC, список функцій якої міститься в Додатку.

Фрейми

Синтаксично *фрейм* можна уявляти собі як *пойменовану асоціативну списковую структуру (ПАСС)*, перший елемент якої є ім'ям фрейму, а решта - його слотами. Ім'я фрейму має бути атомом (в сенсі мови ЛІСП). Кожен фрейм має унікальне ім'я в системі.

Слот в свою чергу є ПАСС, перший елемент якої - ім'я слота. Інші компоненти називаються аспектами. Ім'я слота має бути атомом. Кожен слот має унікальне ім'я усередині фрейму. Порядок слотів всередині фрейму є несуттєвим.

Аспект також є ПАСС. Іменем аспекту служить його перший елемент. Інші елементи називаються даними. Ім'я аспекту (атом) унікально всередині слота. Порядок аспектів в слоті є несуттєвим.

Дане - це ПАСС, перший елемент якої є ім'я даного або його значення. Інші елементи даного називаються коментарями. Даним може бути будь-який об'єкт в сенсі мови ЛІСП (в тому числі ім'я функції або виклик функції), а також фрейм або ім'я фрейма. Кожне значення унікально всередині аспекту. Порядок даних в аспекті є несуттєвим.

Коментар є пойменованим множиною, перший елемент якого називається міткою, а решта - повідомленнями. Мітка повинна бути атомом. Кожен коментар має унікальну мітку всередині даного. Порядок коментарів в даному випадку неважливий.

Повідомленням може бути будь-який об'єкт ЛІСП. Повідомлення вставляються в коментар як в стопку (стек), а видаляються з нього за першим входженню.

Синтаксис фрейма в розширеній нотації Бекуса-Наура (метасимволи: "**::=**" "**<**" "**>**" "**|**" "**{**" "**}**" "**[**" "**]**") має наступний вигляд:

```
<фрейм> ::= ( <ім‟я фрейма> { <слот> } )
     <ім'я фрейма> ::= <атом>
     <слот> ::= ( <ім'я слота> { <аспект> } )
     <ім'я слота> ::= <відношення> |
<властивість> | SELF
     <відношення> ::= <атом>
     <властивість> ::= <атом>
     <аспект> ::= ( <ім'я аспекта> { <данне> }
)
     <ім'я аспекта> ::= <атом>
     <данне> ::= ( <ім'я данного> { <коментар>
} ) | <функтор>
```

```
    <функтор> ::= OR | ALT | NOT | IF | THEN
| ELSE | FI
    <ім'я данного> ::= <значення>
    <значення> ::= <атом> | <d-пара> |
                   <ім'я ЛІСП- или ФРЛ-
функции> |
                   <виклик ЛІСП- або ФРЛ-
функції> |
                   <ім'я фрейма> | <фрейм>
    <комментар> ::= ( <ім'я комментария>
{ <повідомлення> } )
    <ім'я комментар> ::= <мітка>
    <мітка> ::= <атом>
    <повідомлення> ::= <s-вираз>
```

Типи даних

У ФРЛ існують три типи значень (даних): а) пряме значення; б) непряме значення; в) обчислюване значення.

Непряме дане має коментар **(STATUS: INDIRECT)** і може також містити коментарі з мітками **SLOT:** і **FACET :**. Запит на непряме дане з ім'ям **V** з аспекту **A** слота **S** фрейма **F** буде переадресовано у фрейм **V** слот **S** аспект **A**. При цьому повідомлення з коментарів з мітками **SLOT:** і **FACET:** (якщо вони задані) замінюють **S** і **A** відповідно. Якщо ім'ям непрямого даного є *****, то вона позначає ім'я поточного фрейма.

Приклад. Нехай є фрейми F1 і F2:

```
F1: (F1 ... (S ... (A (F2 (STATUS: INDIRECT)
(SLOT: Q) ) )
                                        ... ) ... )
 F2: (F2 ... (S ... (A (V1) )
             (Q ... (A (V2) ) …)
```

Тоді у відповідь на запит на дане з аспекту **A** слота **S** фрейму **F1** буде видано значення **V2**, тому що значення **F2** є непрямим.

Обчислюване дане - це таке дане, яке:

- або міститься в аспектах **$ IF-ADDED**, **$ IF-REMOVED**, **$ IF-NEEDED**, **$ IF-INSTANTIATED**, **$ REQUIRE**, **$ IF-GET**, **$ IF-PUT**, **$ IF-REM** і не має коментаря **(STATUS: NOEVAL)**,

- або міститься в аспектах **$ VALUE**, **$ DEFAULT** і має коментар **(STATUS: EVAL)**.

Будь-яке обчислюваних дане може мати також коментарі з мітками **PARM:** або **PARMQ :**.
Якщо ім'ям вичіслімого даного є атом **P**, то відповіддю на запит на це дане є результат виклику процедури **P** з актуальними аргументами з аспектів з мітками **PARM:** або **PARMQ:** (в останньому випадку аргументи не обчислюються). Якщо ж ім'ям обчислюваного даного є неатомарная ЛІСП-форма, то відповіддю на запит на значення такого даного є результат обчислення цієї форми.

Звернення до приєднаної процедурі має синтаксис:

```
( <ім"я процедури>
    (PARM:  <S-вираз1> <S-вираз2> ... )
    (PARMQ: <S-вираз1> <S-вираз2> ... )
  [(STATUS: EVAL )] )
```

або

```
( <неатомарна ЛІСП-форма> [(STATUS: EVAL)]).
```

У першому випадку процедура з вказаним ім'ям застосовується до списку аргументів, що формується на підставі коментарів, тоді як у другому випадку звичайним способом обчислюється форма ЛІСП.

Зауваження: тут і далі при визначенні синтаксису в квадратні дужки беруться необов'язкові елементи, в фігурні - альтернативні елементи, а в кутові дужки полягають нетермінальні символи.

Приклад. Нехай є фрейм **F**

```
F: (F ... (S ... ($VALUE (LIST (STATUS: EVAL)
                                        (PARMQ:
A B C) ))
                                    ... ) ... )
```

Тоді у відповідь на запит даних з аспекту **$ VALUE** слота **S** фрейма **F** буде видано не дане з ім'ям **LIST**, а дане з ім'ям **(A B C)**, тобто з ім'ям, отриманим в результаті обчислення функції **LIST** з аргументами **A**, **B** і **C**. Те ж саме вийде і в разі, коли фрейм **F** має вигляд

```
F: (F ... (S ... ($VALUE ((LIST 'A 'B 'C)
(STATUS: EVAL)))
                                    ... ) ... )
```

Всі інші дані є прямими. Вони беруться з мережі фреймів без будь-якої додаткової обробки.
Кожен запит на дані у фрейм **F** слот **S** аспект **A** виконується в спеціальному ФРЛ-середовищі, в якому:

- системна змінна: **FRAME** пов'язана з **F**;
- системна змінна: **SLOT** пов'язана з **S**;
- системна змінна: **FACET** пов'язана з **A**.

Типи коментарів

У ФРЛ є кілька типів коментарів:

- коментар з міткою **STATUS:** використовується для індикації методу обробки даного при його витяганні. Можливими повідомленнями для цього коментаря є **EVAL**, **NOEVAL** і **INDIRECT**;

- коментарі з мітками **SLOT:** і **FACET:** використовуються для вказівки референта непрямого даного. Вони використовуються спільно з коментарем **(STATUS: INDIRECT)**;

- коментарі з мітками **PARM:** і **PARMQ:** використовуються при завданні аргументів для приєднаних процедур;

- коментар з міткою **FINHERIT:** використовується для локального управління успадкуванням. Якщо повідомлення є **CONTINUE**, то містить його слот успадковує дані зі своїх АКО-прототипів навіть в разі, якщо він сам і містить дані, які розшукуються. Повідомлення **STOP** забороняє успадкування даних, шуканих в містить його слоті;

- коментар **(TYPE: <тип>)** забезпечує можливість виборчого виклику процедур функціями **FPROC**, **FNEED** і **FEXEC**;

- коментар з міткою **IN**: вставляється системою в дані при їх вилученні для ідентифікації фрейму, слоти і аспекту, з яких вони були вилучені.

Модифікація мережі фреймів

Для модифікації мережі фреймів використовуються функції створення та видалення фрагментів мережі фреймів. Описи цих функцій приведені в Додатку до лекції. Основними функціями цього класу є **FPUT**, **FREMOVE** і **FINSTANTIATE**.

Функція

(FPUT <ім'я фрейма> <ім'я слота> <ім'я аспекту> <значення>

[<Мітка>

[<повідомлення>]])

заносить значення (з міткою і повідомленням, якщо вони задані) за вказаним шляхом, створюючи при необхідності відсутні компоненти (фрейм, слот, аспект). При цьому успадковуються і запускаються процедури, приєднані до аспекту **$IF-ADDED** відповідного слоту цього фрейму або його прототипів.

Функція
(FREMOVE <ім'я фрейма> [<ім'я слота> [<ім'я аспекту> [<значення> [<мітка> [<повідомлення>]]]]])
видаляє з фрейма подструктуру, зазначену відповідним шляхом. При цьому, якщо видаляється значення, то успадковуються і запускаються процедури, приєднані до аспекту **$IF-REMOVED** містить слота або успадковані по АКО-ієрархії.

Функція
(FINSTANTIATE <ім'я прототипу> [<ім'я екземпляра>] [<слот>])
створює екземпляр фрейма-прототипа і заповнює цей екземпляр зазначеними слотами. При цьому успадковуються і виконуються процедури, приєднані до аспектам **$ IF-INSTANTIATED** всіх слотів фрейму-прототипу і отримані результати заносяться в аспекти **$DEFAULT** відповідних слотів фрейма-екземпляра.

При створенні баз фреймів часто доводиться створювати реверсивні зв'язки. Для автоматизації цього процесу в ФРЛ передбачені функції **FREVADD**, **FREVREM**, **FORWADD** і **FORWREM** (див. Додаток).

Наприклад, процедура **FREVADD**, приєднана до аспекту **$IF-ADDED** слота **S** фрейма **F**, створює реверсивні зв'язку у всіх його фреймах-примірниках при занесенні в них прямих зв'язків:

- стан бази фреймів до встановлення між фреймами **F1** і **F2** зв'язку **S**:

```
F: (F  ... (S   ($IF-ADDED ((FREVADD 'SREV))))
... )
     F1: (F1 ... (AKO ($VALUE     (F) ) ) ... )
     F2: (F2 ... (AKO ($VALUE     (F) ) ) ... )
```

- стан бази фреймів після встановлення зв'язку **S** між фреймами **F1** і **F2**:

F: без змін

```
F1: (F1 ... (AKO   ($VALUE (F) ) )
            (S     ($VALUE (F2) ) ) ... )
F2: (F2 ... (AKO   ($VALUE (F) ) )
            (SREV ($VALUE  (F1) ) ) ... )
```

Трасування

У ФРЛ є можливість трасування, тобто виконання певних програм користувача, при внесенні змін до будь-якої фрейм, слот, аспект, дане, коментар і повідомлення. Для цієї мети в системі є спеціальні засоби, через які здійснюється вплив користувача на роботу ФРЛпроцессора. До таких засобів відноситься процедура **FTRACE**. За допомогою **FTRACE** користувач може підключати до системи процедури (будемо називати їх трасуючими), які керують запуском приєднаних процедур.

Трасуючі процедури можуть бути приєднані до будь-якого кадру із заданою структурою. Для підключення цих процедур до роботи ФРЛ-процесора необхідно виконати функцію: **(FTRACE <ім'я-фрейма> <список-пар>)**, де **<ім'я-фрейма>** - ім'я фрейму, що містить трасуючі процедури, а **<список-пар>** - список виду **((<мета-ім'я-підструктури> <умова> ...)**.

<Мета-ім'я-підструктури> - це є **FRAME**, **SLOT**, **VALUE**, **LABEL** або **MESSAGE**.

<Умова> представляє один з наступних атомів: **IF-ADDED**, **IF-REMOVED**, **IF-GETED**, що інтерпретується наступним чином:

IF-ADDED - трасуючі процедури запускаються при додаванні в систему структури з відповідним мета-ім'ям;

IF-REMOVED - трасуючі процедури запускаються при видаленні з системи структури з відповідним мета-ім'ям;

IF-GETED - трасуючі процедури запускаються при вилученні з мережі структури з відповідним мета-ім'ям.

Можна використовувати і таке звернення: **(TRACE <ім'я фрейма>)** . В цьому випадку будуть об'явлено трасуючими всі процедури з фрейма **<ім'я фрейма>**.

Для відключення режиму трасування слід використовувати функцію **FUNTRACE**. Синтаксис: **(FUNTRACE)**. Нижче наводиться структура фрейму, використовуваного процедурою **FTRACE**.

```
(<Ім'я фрейма>
      (FRAME ($ IF-ADDEED <процедури, що
виконуються при занесенні нового фрейму в
систему>)
             ($ IF-REMOVED <процедури, що
виконуються при видаленні кадру з системи>)
             ($ IF-GETED <процедури,
извлекающие інформацію з фрейму>)
      )
```

```
(SLOT ($ IF-ADDED <процедури, що виконуються
при занесенні нового слота в який-небудь
фрейм>)
            ($ IF-REMOVED <процедури, що
виконуються при видаленні слота з будь-якого
фрейму>)
            ($ IF-GETED <процедури, извлекающие
інформацію з слота>)
      )
```

```
(FACET ($ IF-ADDED <процедури, що виконуються
при занесенні нового аспекту в будь-якої
слот>)
        ($ IF-REMOVED <процедури, що
виконуються при видаленні аспекту з будь-якого
слота>)
        ($ IF-GETED <процедури, извлекающие
значення з аспекту>)
     )
```

```
(VALUE ($ IF-ADDED <процедури, що виконуються
при занесенні
                        нового значення в
будь-який аспект>)
            ($ IF-REMOVED <процедури, що
виконуються при видаленні
                        значення з будь-
якого аспекту>)
            ($ IF-GETED <процедури,
извлекающие значення з
                        даного>)
     )
```

```
(LABEL ($ IF-ADEED <процедури, що виконуються
при занесенні
                    нового коментаря в якесь
дане>)
            ($ IF-REMOVED <процедури, що
виконуються при видаленні
                        коментаря з будь-
якого даного>)
            ($ IF-GETED <процедури,
извлекающие коментар з
                    будь-якого даного>)
     )
```

```
(MESSAGE ($ IF-ADDED <процедури, що
виконуються при занесе-
                              нні нового
повідомлення в будь-якої
                              коментар>)
                 ($ IF-REMOVED <процедури, що
виконуються при удаленні повідомлення з будь-
якого коментаря>)
      )
   )
```

Зауваження: Всі компоненти зазначеної фрейм-структури носять факультативний характер.

Взаємодія з віртуальної базою об'єктів

У ФРЛ застосовується розподіл фреймів на активні і пасивні. Активними називаються фрейми, що знаходяться в оперативній пам'яті, як правило, це фрейми, створені в поточному сеансі роботи або завантажені з зовнішніх запам'ятовуючих пристроїв (ВЗП). Фрейми, котрі розташовані у ВЗУ, називаються пасивними.

На ВЗУ фрейми можна зберігати або в послідовному файлі, або у віртуальній базі об'єктів (ВБО). Під ВБО розуміється інформаційний набір на ВЗУ з можливістю асоціативного поіменного доступу до своїх об'єктів, що мають списковую структуру.

Перевагою зберігання фреймів або процедур в послідовному файлі є можливість їх редагування будь-яким потужним системним редактором, недоліком - складність і незручність динамічного взаємодії з ФРЛ.

Перевагою зберігання об'єктів в ВБО є можливість завантаження і вивантаження цих об'єктів за розпорядженням тільки їх імен (при цьому немає необхідності знати імена файлів, в яких зберігаються ці об'єкти). При роботі з ВБО забезпечується автоматичне завантаження фреймів і процедур в оперативну пам'ять. Недоліком є відсутність можливості використання будь-якого системного редакто-
ра за винятком вбудованого редактора системи Фориси. Процес активації пасивних фреймів (завантаження їх з ВЗП в оперативну пам'ять) може проходити або за явною вимогу користувача з конкретного файлу або автоматично з ВБО. Деактивация активних фреймів і / або процедур збереження в ВБО або в файлі проводиться тільки за явною вказівкою користувача. Як активація, так і деактивація можливі лише за умови, що Вам потрібно переглянути бази даних або послідовний файл попередньо відкритий.

ОРГАНІЗАЦІЯ МЕРЕЖ ФРЕЙМІВ

Мережі на фреймах можуть створюватися за допомогою використання імен фреймів як значення слотів. Найпростіша зв'язок з ім'ям **R** від фрейму **F1** до кадру **F2** виглядає так:

Фрейм **F1: (F1 ... (R ($ VALUE (F2))) ...)**

У ФРЛ існують два спеціальних типи зв'язків між фреймами:

- **AKO**-зв'язок (від англійського "A Kind of" - "один з");
- **INSTANCE**-зв'язок (зворотна по відношенню до **AKO**).

Ці зв'язки служать для створення **AKO / INSTANCE**-ієрархій. Нехай фрейм **F1** пов'язаний **AKO**-зв'язком з фреймом **F2**. В цьому випадку фрейм **F2** є прототипом для фрейма **F1**, а фрейм **F1** - екземпляром для **F2**. Кожен фрейм може мати кілька прототипів і примірників.

Якщо фрейм **F1** з'єднаний **AKO**-зв'язком. з фреймом **F2**, то все значення фрейму **F2** можуть успадковуватися фреймом **F1**. Це означає, що кожен запит на значення деякого гнізда під фрейм **F1** може бути доповнений запитом на значення цього ж слота, але у фрейм **F2** і т.д.

Термін "спадкування даних" (= "успадкування значень властивостей", = "успадкування властивостей") в сенсі ФРЛ означає "заміну запиту до деякого кадру за бажаними даними запитом до (може бути) іншого фрейму за (може бути) іншими даними". У цьому визначенні відображена важлива особливість спадкування властивостей в ФРЛ: успадкування властивостей є суто динамічним поняттям і є тільки в момент пошуку відповіді на запит.

У ФРЛ є три типи успадкування властивостей, що зберігаються в аспекті **$VALUE**:

- **AKO**-успадкування;
- локальне спадкування;
- **DEFAULT**-успадкування.

Для властивостей з інших аспектів валідності тільки **AKO**-успадкування. Термін "AKO-успадкування" означає "заміну запиту до деякого кадру за даними запитом до його прототипу за цими даними".
Термін "локальне спадкування" означає "заміну запиту до деякого кадру за даними запитом до його локального прототипу за цими даними". Фрейм **F1** називається локальним прототипом для фрейма **F2** щодо слота **S**, якщо він знаходиться серед значень у фреймі **F2** слоті **S** аспекті **$AKO**:

```
(F2 ... (S ... ($AKO ...
(F1) ... ) ... ) ... )
```

Локальне спадкування більш пріоритетно, ніж глобальне.
Термін "**DEFAULT**-успадкування" означає "заміну запиту до деякого кадру за даними запитом до цього ж кадру за цими ж даними, але з аспекту **$DEFAULT**".

Приклад **AKO**-успадкування. Нехай дано фрейми **F1** і **F2**:

```
F1: (F1 ... (S ... ($VALUE (V1)
(V2)) ... ) ... )
      F2: (F2 ... (AKO ($VALUE (F1))) ... )
```

Тоді у відповідь на запит до кадру **F2** за значеннями слота **S** будуть видані значення **V1** і **V2**.

Зауважимо, що **AKO**-ієрархія (як глобальна так і локальна) може бути імпліцитної, тобто будуватися динамічно в процесі пошуку. Для цього дане в слоті **AKO** фрейма-екземпляра (в прикладі це **F2**) має бути обчислюваності або непрямим.

ВИКОРИСТАННЯ ПРИЄДНАНИХ ПРОЦЕДУР

У ФРЛ існують два способи використання (активації) приєднаних процедур:
а) явний виклик; б) виклик по ситуації.
При явному виклику вказуються імена фрейму, слоти і аспекту (як правило, це **`$IF-NEEDED`**), в яких міститься виклик необхідної процедури. Функції ФРЛ, керуючі явним викликом приєднаних процедур, наведені в Додатку.

При ситуативному виклику вказується умова, при виконанні якого автоматично викликається необхідна процедура. У ФРЛ є такі стандартні ситуації: **$IF-ADDED**, **$IF-REMOVED**, **$IF-INSTANTIATED**, **$IF-GETED**, **$IF-ADD**, **$IF-PUT**, **$IF-REM**, **$IF-GET**. У всіх цих випадках приєднані процедури зберігаються під однойменними аспектами і можуть успадковуватися.

Якщо до одного аспекту приєднано кілька процедур, можливо розділених функторами, то вони обчислюються відповідно до таких правил:

- у відсутності функторів відбувається послідовний виклик процедур за допомогою **FAPPLY**, причому результатом є кон'юнкція результатів окремих процедур. Це означає, що перша зустрінута процедура зі значенням **NIL** припиняє процес.

- функторами **ALT** поділяються альтернативи. Якщо обчислення першої альтернативи дає значення **NIL**, то обчислюється наступна і т.д.

- функторами **OR** поділяються альтернативи всередині окремого Кон'юнктів.

- функтор **NOT** означає, що значення наступного за ним процедури слід поміняти на протилежний, тобто НЕ **NIL** на **NIL**, а **NIL** - на **T**.

- функтор **IF** означає початок умовної конструкції, що має традиційний вигляд:

<Ум. констр.> :: = IF <послідовність кон'юнктів>
THEN <послідовність кон'юнктів>
[ELSE <послідовність кон'юнктів>] FI

причому **<послідовність кон'юнктів>** може містити всі функтори, крім **ALT**.

При обчисленні приєднаних процедур актуальне ФРЛ-середовище. Крім цього, змінна: **VALUE** приймає в якості значення то дане, яке викликає її активізацію.

Процедури з аспекту **$IF-ADDED** викликаються в тому випадку, якщо в якому їх слот додається нове дане.

Приклад. Нехай є фрейми **F1** і **F2**:

```
F1:  (F1  ...  (S  ($IF-ADDED  ((PUSH
(LIST  :FRAME :SLOT
                                    :VALUED)
*ADDED*)))) ... )
        F2:  (F2  ...  (AKO ($VALUE (F1))) ...
)
```

Тоді при занесенні значення **5** в слот **S** фрейма **F2** спрацює процедура **PUSH**, яка занесе в стек *** ADDED *** список **(F2 S 5)**.

Процедури з аспекту **$IF-REMOVED** викликаються при видаленні даного з містить їх слота.

Приклад. Нехай є фрейми **F1** і **F2**:

```
F1: (F1 ... (S ($IF-REMOVED ((PUSH :VALUE
*REMOVED*))))

... )
```

```
F2: (F2 ... (AKO ($VALUE (F1)))
            (S ($VALUE (5))) ... )
```

Тоді при видаленні значення **5** з слота **S** фрейма **F2** спрацює процедура **PUSH**, яка занесе **5** в стек *** REMOVED ***.

Процедури з аспекту **$IF-INSTANTIATED** виконуються в разі створення екземпляра деякого фрейма (див. Опис функції **FINSTANTIATE** в Додатку).

Процедури з аспекту **$IF-GETED** викликаються при видаленні даного з містить їх слота.

Аспекти **$ IF-GET**, **$ IF-ADD**, **$ IF-REM** у цій версії передбачені тільки в функціях: **FGET-**, **FGET1-**, **FGETN** і містять процедури конвертації даних, призначення яких полягає в наступному: над кожним з даних ФРЛ, що беруть участь в операціях створення, видалення, пошуку, виконуються процедури з аспектів **$IF-ADD** (при додаванні даного у фрейм), **$IF-REM** (при видаленні даного з фрейма), **$ IF-GET** (при добуванні даного з фрейма). При цьому дане ФРЛ замінюється на результат виконання цієї процедури. Якщо результат **NIL**, то над цими даними ніяких операцій не проводиться.

Аспекти конвертації надають користувачеві потужні засоби, що дозволяють з кожним даними пов'язувати відповідну обробну процедуру. Ця можливість представляє інтерес при розширенні стандартних можливостей ФРЛ (розширення механізмів успадкування, коментарів і т.п.). Наприклад, на кадрі **F** слоті **S** імена даних є числами і атомами.

```
(F
        (S ($VALUE (5) (A) (6) (B)))
    )

    (FGET 'F 'S) => (5 A 6 B)
```

Для того щоб функція **FGET** з слота **S** фрейма **F** витягувала тільки числа, фрейм **F** слід модифікувати таким чином:

```
(F
        (S ($VALUE (5) (A) (6) (B))
           ($IF-GET ((AND
(NUMBERP :VALUE) :VALUE)))
        )
    )
```

Після модифікації:

```
(FGET 'F 'S) => (5 6)
```

Процедури з аспекту **$IF-INSTANTIATED** виконуються в разі створення екземпляра деякого фрейма.

СТРАТЕГІЇ ПОШУКУ ДАНИХ НА МЕРЕЖАХ ФРЕЙМІВ

Для отримання даних з мережі фреймів в ФРЛ є набір спеціальних функцій (див. Додаток), основною з яких є функція **FGET**.

У мові ФРЛ є засоби контролю як стратегії пошуку даних, так і бажаного виду відповіді на запит. Для цієї мети є спеціальні керуючі ключі, які використовуються у функціях вилучення інформації.

Функція

(FGET <ім'я фрейма> <ім'я слота> [<ім'я аспекту> [<ключі>]])

отримує дані з вказаного аспекту зазначеного слота зазначеного фрейму. Якщо аспект не заданий, то за замовчуванням він дорівнює **$VALUE**. Форма результату і стратегія пошуку задаються за допомогою ключів. Операнд ключі є список, елементами якого можуть бути: **E, C, - !, - @, - *, O, -H, 0, 2**. Ці ключі мають наступний сенс:

E - результат є першим витягнуте дане;

C - дані в результаті будуть мати коментарі;

-! - обчислюваних дані не будуть обчислюватися (тобто будуть оброблятися як прямі);

- **@** - непрямі дані обробляються як прямі;
- ***** - забороняє обробку ***** як імені поточного фрейма при

використанні в непрямому даному;

Про-дані успадковуються з першого зустрінутого релевантного фрейму на кожному **AKO**-шляху;

-H - забороняє **AKO**-успадкування;

0 (нуль) - забороняє **DEFAULT**-успадкування;

2 - забороняє **DEFAULT**-успадкування, але в разі неуспішного пошуку даних в аспекті **$VALUE** повторює той же запит, але в аспект **$DEFAULT**.

За замовчуванням діють наступні ключі:

L - результат виглядає як список всіх витягнутих даних;

-C - кожне дане в результаті буде представлено тільки своїм ім'ям, тобто результатом буде не список даних, а список значень;

! - кожне обчислюваності дане буде обчислено;

@ - кожне непряме дане буде замінено своїм референтом;

***** - символ "*****" на місці значення буде замінений ім'ям поточного фрейма;

A - дані успадковуються з усіх релевантних фреймів;

H - **AKO**-успадкування дозволено;

1 - **$DEFAULT**-успадкування дозволено.

Механізми успадкування реалізуються функціями **FGET, FGET1, FGETN**.

FGET1 - тобто наслідування уздовж **AKO** зв'язку, причому при виявленні першого референта в мережі фреймів уздовж **AKO** зв'язку процес успадкування припиняється.

FGETN - спадкування здійснюється за всіма фреймам уздовж **AKO** зв'язку.

Ниже наведено алгоритм роботи **FGET1** (для **FGET** п.3 алгоритму відсутній, для **FGETN** п.3 розширено можливістю обробки всієї множини запитів для **AKO**).

1. Якщо надійшов запит у фрейм :**FRAME**, слот :**SLOT**, аспект **$VALUE** і дані виявлені, то перейти до п.4, інакше до п.2.

2. Виконати запит у фрейм :**FRAME**, слот :**SLOT**, аспект **$DEFAULT**. Якщо дані виявлені, то перейти до п.1 і послідовно виконувати запити у фрейм :**FRAME**, причому :**FRAME** буде кожен раз зв'язуватися з черговим даними, отриманим на даному етапі 3 до тих пір, поки не буде задоволено запит. В іншому випадку перейти до п.4.

3. Результат запиту **NIL**. Перейти до п.5.

4. Список обраних даних зв'язати зі значенням змінної :**VALUE**, виконати приєднані процедури з аспекту **$IF-GETED** за правилами, розглянутими в п.3 (виконання приєднаних

процедур будемо називати надалі побічним ефектом запиту).
Результат запиту - список обраних даних. Перейти до п.6.

5. Кінець роботи.

Примітки:

1) Вищеописаний алгоритм не включає опис того впливу на вибір даних при виконанні запиту, який чинять коментарі та повідомлення, приєднані до даних з аспектів **$VALUE** і **$DEFAULT** (надалі будемо називати цей вплив редагуванням або фільтрацією запиту).

2) Якщо в запиті **:FACET** відмінний від **$ VALUE** або **$ DEFAULT**, то виконується звичайна вибірка даних без успадкування і побічних ефектів.

3) Існують функції **FGET-, FGET1-, FGETN-**, які аналогічні **FGET, FGET1, FGETN**, але без побічного ефекту і фільтрації запиту.

Функція
(FGET-SEL <ім'я фрейма> <ім'я слота> <ім'я аспекту> [<ключі>]

{<Мітка> <повідом>})
працює так само, як і **FGET**, але витягує тільки ті дані, які містять зазначені в зверненні коментарі.

Функція

(FGETIND <ім'я фрейма1> ({<ставлення>})
[<ім'я аспекту>
[<Ключі>]])

здійснює вилучення даних листя фреймів з дерева фреймів, пов'язаних зазначеними відносинами. Більш точно: нехай є дерево фреймів виду:

(F0 R1 (F1,1 ...) ... R1 (F1, N1))

де **FI, J** - фрейми **(0 <I <K-1, 1 <J <NI)**;

FK, J - будь-які дані в сенсі MFRL / PC (в тому числі і фрейми) **(1 <J <NK)**.

Тоді в результаті виконання **(FGETIND F0 (R1 R2 ... RK))** буде отримано результат **(FK, 1 ... FK, NK)**. За замовчуванням передбачається, що всі дані шукаються в аспекті **$VALUE**.

Функція

(FQUERY <f - ім'я фрейму> <s - ім'я слота> <a-ім'я аспекта>
<Ref - критерій> <field - область пошуку>
<Relation - відношення>)

здійснює пошук на мережі фреймів. Перед початком пошуку визначаються: область пошуку (множина об'єктів-кандидатів), об'єкт пошуку та критерій пошуку.

Основні випадки:

1. **f = NIL & ref <> NIL**

Область пошуку визначається наступним чином:

Якщо **`field = NIL`**, то область пошуку - за всіма активними фреймам.

Якщо **`field`** - атом, то по всіх вершин дерева відносини **`relation`** з коренем в **`field`**.

Якщо **`field`** - непорожній список, то якщо **`relation`** задано — по всіх вершинах всіх дерев відносини **`relation`** з корінням в списку **`field`**; якщо **`ж relation`** не задано, то за всіма елементами списку **`field`**.

Об'єкт пошуку - фрейм.

Критерій пошуку наступний:

Якщо **`ref`** - атом, то він повинен міститися серед значень слота **`s`** аспекту a фрейма-кандидата.

Якщо **`ref`** - предикат, то він повинен виконуватися хоча-б на одному значенні слота **`s`** аспекту a фрейма-кандидата.

Результат - список імен фреймів.

2. **`f <> NIL & s = NIL`**

Область пошуку - всі слоти фрейму **`f`**.

Об'єкт пошуку - слот.

Критерій пошуку - в аспекті **`a`** слота-кандидата фрейма **`f`** має міститися дане, яке задовольняє **`ref`** (аналогічно попередньому випадку - див. Вище).

Результат - список імен слотів.

3. **`f <> NIL & s <> NIL & a = NIL`**

Результат - структура слота **`s`** фрейма **`f`**.

4. **`f <> NIL & s <> NIL & a <> NIL`**

Область пошуку - всі дані фрейма **`f`** слота **`s`** аспекту **`a`**.

Об'єкт пошуку - це або истинностное значення.

Результат пошуку наступний:

- якщо **`ref`** - атом, то **`T`** тільки в тому випадку, коли цей атом **`ref`** міститься серед значень слота **`s`** фрейма **`f`**.
- якщо **`ref`** - предикат, то список всіх даних, на яких цей предикат правдивий.

ПОРІВНЯННЯ ФРЕЙМІВ

Існує безліч різних понять метчінг: строковий, списковий, структурний, синтаксичний, семантичний і т.д. Ми розглядаємо метчінг фреймів, який є структурним і семантичним. За основу взято визначення метчінг з [1]: "фрейм **F1** можна порівняти (або метч) з фреймом **F2**, якщо він може бути його примірником, тобто фрейм **F1** не містить таких властивостей, які суперечили б відомим властивостям фрейма **F2**". Однак в цьому "визначенні" не зовсім ясно, що означає "протиріччя властивостей", тому будемо дотримуватися власного визначення метчінг на фреймах, яке наводиться нижче.

Дескриптори

У метчінгу можуть брати участь об'єкти трьох типів: терми, фрейми і дескриптори.

Об'єкти, які не є фреймами або дескрипторами, вважаються термами.

Об'єкт типу дескриптор має структуру, подібну до структури фрейму, однак на рівні слотів і значень слотів він може мати альтернативні, кон'юнктівние, диз'юнктивні і негативні елементи. Крім цього, значенням слота в свою чергу може бути дескриптор.

Синтаксис дескриптора:

```
<Дескриптор> :: = (<ім'я дескриптора>
{<слот>})
     <Ім'я дескриптора> :: = <атом>
     <Слот> :: = (<ім'я слота> {<аспект>}) |
<Функтор>
     <Функтор> :: = ALT | OR | NOT
     <Ім'я слота> :: = <ставлення> |
<Властивість> | SELF
     <Ставлення> :: = <атом>
     <Властивість> :: = <атом>
     <Аспект> :: = (<ім'я аспекту> {<дане>})
     <Ім'я аспекту> :: = <атом>
     <Дане> :: = (<ім'я даного> {<коментар>})
|
                          <Функтор>
```

```
<Ім'я даного> :: = <значення>
<Значення> :: = <атом> | <D-пара>
              | <Виклик ЛІСП- або MFRL / PC-
функції>
              | <Ім'я фрейма> | <Фрейм>
              | <Ім'я дескриптора> |
<Дескриптор>
<Коментар> :: = (<ім'я коментаря>
{<повідомлення>})
<Ім'я коментаря> :: = <мітка>
<Мітка> :: = <атом>
<Повідомлення> :: = <з-вираз>
```

Дескриптор і аспект дескриптора можуть мати альтернативи. Альтернативою в дескрипторі (або в аспекті) називається мінімальна група слотів (або значень), обмежена функторами **ALT** або межами дескриптора (або аспекту). Альтернатива, в свою чергу, являє собою КНФ, де окремі Кон'юнктів представляються або у вигляді слотів (або значень), можливо, з запереченнями, тобто функторами **NOT**, або у вигляді диз'юнкції слотів (або значень), також можливо з запереченнями. Диз'юнкт в Кон'юнктів поділяються функторами **OR**. Приклад дескриптора наведено далі. Пріоритети функторів наступні: заперечення - 4, диз'юнкція - 3, кон'юнкція - 2, альтернатива - 1.

Основною функцією MFRL / PC, яка здійснює процес зіставлення, є функція **FMATCH?** . Для роботи з об'єктами типу дескриптор передбачені спеціальні функції **DEDESCR**, **DEDESCRQ, DESCRS, DNAME ?, DESCR, DINSTANCE**, синтаксис яких аналогічний відповідно синтаксису функцій **DEFRAME, DEFRAMEQ, FRAMES, FNAME ?, FRAME** і **FINSTANCE** (див. Додаток).

Визначення метчінга

Об'єкт **F1** можна порівняти (або Метч) з об'єктом **F2** (позначення: **F1 M F2**), якщо виконано одну з умов (через **x *** позначається екземпляр об'єкта **x**, що породжується в процесі метчінг):

а) **F1** - фрейм, **F2** - фрейм, причому фрейм **F1** є **AKO**-екземпляром фрейма **F2**.

б) **F1** - фрейм, **F2** - фрейм, причому фрейм **F1** не є **AKO**-екземпляром фрейма **F2** і для кожного значення **V2** будь-якого слота **S** фрейма **F2** може бути знайдено значення **V1** в однойменному слоті **S** фрейма **F1**, яке з ним можна порівняти, тобто **V1 M V2**.

в) **F1** - фрейм, **F2** - дескриптор, причому **F1** можна порівняти (в сенсі умови б) хоча б з однією альтернативою фрейма **F2**. Процесом зіставлення фрейму з альтернативою дескриптора керують функтори.

г) **F1** - фрейм, **F2** - терм, причому **F2** збігається зі значенням слота **SELF** фрейма **F1**, тобто **F2 = (FGET F1 'SELF' $ VALUE '(E))**

д) **F1** - терм, **F2** - фрейм, причому терм **F1** задовольняє предикатам з аспекту **$REQUARE** слота **SELF** фрейма **F2**, тобто **(FCHECK? F2 'SELF F1) = T**

е) **F1** - терм, **F2** - дескриптор, причому терм **F1** задовольняє предикатам з аспекту **$REQUARE** слота **SELF** примірника дескриптора **F2**, тобто **(FCHECK? F2 * 'SELF F1) = T**.

ж) **F1** - терм, **F2** - терм, причому **F1 = F2**.

Зауважимо, що з порожнім фреймом або дескриптором зіставляється будь-який об'єкт.

Слот **SELF** використовується в кінцевих точках процесу зіставлення, коли необхідно зіставити терм і нетерм. Він як би представляє весь фрейм в цілому.

Результат зіставлення

Результатом зіставлення є істина (тобто атом **T**) в випадку успішного зіставлення і брехня (тобто атом **NIL**) - в разі неуспішного.

Однак, в результаті зіставлення, як правило, необхідно знати не тільки бінарний відповідь: зіставити - непорівнянний, але і причини цієї відповіді, тобто які слоти і значення зіставити або непорівнянний. Для цієї мети в системі MFRL / PC введені глобальні змінні ***MATCHED*** і ***UNMATCHED***.

В змінної ***MATCHED*** формується список пар зіставити об'єктів, причому група пар, які стосуються одного слоту, передує ім'ям цього слоту.

В змінної ***UNMATCHED*** формується список пар непорівнянний об'єктів, причому група пар, які стосуються одного слоту, також передує ім'ям цього слоту.

Приклад. Нехай дано фрейм **F** і дескриптор **D**:

```
F: (F (S1 ($VALUE (V1) (V2)))
                  (S2 ($VALUE (V1) (V3)))
                  (S3 ($VALUE (V2)
(V3))) )

                D: (D (S1 ($VALUE (V1) (V2)))
                      (S2 ($VALUE (V1) OR
(V2)))
                      (S3 ($VALUE (V2) NOT
(V3)))
                  ALT (S1 ($VALUE (V3) OR
(V1)))
                      (S2 ($VALUE (V2)))
                   OR (S3 ($VALUE (V3))) )
```

Тоді результатом метчінгу **F M D** буде **T**, причому:

```
*MATCHED* = ((F . D)
                        S3 (V3 . V3)
                        S1 (V1 . V1) (V3 . V3) (V2
. V2)
                        S2 (V1 . V1)
                        S1 (V2 . V2) (V1 . V1) )
*UNMATCHED* = ((V2 . V3)
               C2 (V3 . V2) (V1 . V2) (V2 . V3)
(V1 . V3)
               C3 (V2 . V3) (V1 . V2))
```

Нижче приведена та частина дескриптора D, з якої успішно зіставити фрейм F:

```
(D (S1 ($VALUE (V3) OR (V1)))
                        (S3 ($VALUE (V3))) )
```

Пропонований формалізм дескрипторів з функторами дозволяє на практиці, наприклад, при створенні експертних систем, реалізовувати в правилах виведення цілком достатній для експерта клас умов.

ВИСНОВОК НА ФРЕЙМАХ

Для представлення логічних закономірностей в мові ФРЛ зручно використовувати приєднані процедури, причому можна комбінувати прямий і зворотний типи виведення, поміщаючи відповідні процедури в аспекти **$IF-ADDED** і **$IF-NEEDED** відповідно.

Для ілюстрації вищесказаного розглянемо фрагмент фрейма відношення "**НА**":

```
(fassertq НА
      ..............
  (REQUIRMENTS ($if-needed ((<= (!! :frame
'АКТ1 'SIZE)
                                (!! :frame
'АКТ2 'SIZE))))))
  (R-RELATION  ($if-added ((progn
        (check-&-add-rel '(НА :akt2 >x)
'(НАД :akt1 >x))
        (check-&-add-rel '(РЯДОМ :akt2 >x)
'(РЯДОМ :akt1 >x))
        (check-&-add-rel '(РЯДОМ >x :akt1)
'(РЯДОМ >x :akt2))
        ))
```

))

. . . . . .

)

В аспекті **$IF-NEEDED** слота **REQUIRMENTS** міститься процедура, яка порівнює розміри (значення властивостей "**SIZE**") наповнювачів ролей **AKT1** і **AKT2** для примірника відносини "**НА**", тобто тут представлено правило псевдофізіческой логіки простору: "Більший об'єкт не може перебувати **НА** меншому об'єкті ". Аналогічно можна уявити фрейми просторових відносин "**В**", "**НАД**" і ін. В ПМ-формі ці знання означають:

"Більший або рівний об'єкт не може перебувати В меншому або рівному об'єкті"

"Менший об'єкт не може перебувати **ПІД** великим об'єктом" і т.д.

Для уявлення продукцій виду

```
(a R1 b) & (b R2 c) => (a R3 c)
```

можна використовувати аспект **$IF-ADDED**. У наведеному вище прикладі ставлення "**НА**" таким чином представлені правила:

```
(a НА b) & (b НА c)     => (a НАД c)
(a НА b) & (b ПОРУЧ c) => (a ПОРУЧ c)
(a НА b) & (a ПОРУЧ c) => (b ПОРУЧ c)
```

Функція **(Check - & - add-rel: rel1: rel2)** перевіряє (за допомогою МАРС), міститься серед примірників відносини: **rel1** (вони перераховані в слоті **R-RELATION**) хоча-б один, який можна порівняти (за допомогою **RMATCH**) з: **rel1**. Якщо так, то в БФ заноситься (за допомогою **RASSERT**) екземпляр відносини: **rel2**. У нашому випадку вона перевіряє, чи міститься серед примірників відносини "**НА**" хоча-б один, який можна порівняти з дескриптором **(НА: akt2> x)**. Якщо так, то в базу фактів заноситься екземпляр відносини "**НАД**".

```
(defun CHECK-&-ADD-REL (:rel1 :rel2)
  (mapc '(lambda (:rel3 :rel4)
           ((RMATCH :rel1 :rel3)
(RASSERT :rel2)) )
         (fget- (car :rel1) 'R-RELATION
$value) )
)
(passertq RMATCH (:r1 :r2) (lmatch (runif :r1)
(funif :r2)))
(passertq lmatch (:l1 :l2)
  ((null :l1) ((null :l2)))
  ((neql (substring (car :l1) 0 0) '>)
    ((eql (car :l1) (car :l2)) (lmatch
(cdr :l1) (cdr :l2))))
  (set (car :l1) (car :l2))
  (lmatch (cdr :l1) (cdr :l2))
```

```
)
(passertq runif (:rel) (cons (car :rel)
(runif1 (cdr :rel))))
(passertq runif1 (:rel)
  (mapcar '(lambda (:arg)
      ((eql (substring :arg 0 0) ':)
         (nth (read-from-string
(substring :arg 4 4)) :value) )
      ((eql (substring :arg 0 0) '>) :arg) )
    :rel)
)
(passertq funif (:fr) (funif1 :fr 1))
(passertq funif1 (:fr :n :w)
  ((setq :w (fgete :fr (pack* 'akt :n)))
    (cons :w (funif1 :fr (add1 :n))) )
)
```

```
(passertq RASSERT (:rel :w)
  (setq :w (fgename (car :rel)))
  (setq :rel (cons (car :rel) (rassert1 :w
(cdr :rel) 1)))
  (fput :w AKO $value 'НАХОДИТСЯ)
  (fput :w 'RELATION $value (car :rel))
  (fput (car :rel) 'R-RELATION $value :rel)
)
(passertq rassert1 (:fr :rel :n :rr)
  ((null :rel) nil)
  ((eql (substring (car :rel) 0 0) ':)
    (fput :fr (pack* 'akt :n) $value
       (setq :rr (nth (read-from-string
              (substring (car :rel) 4
4)) :value)) )
    (cons :rr (rassert1 :fr (cdr :rel)
```

```
(add1 :n))) )
  (fput :fr (pack* 'akt :n) $value (setq :rr
(eval (car :rel))))
  (cons :rr (rassert1 :fr (cdr :rel)
(add1 :n)))
)
```

Допоміжна функція **(!!: fr: akt: prop)** видає значення **$value** слота **:prop** від слота **:akt** фрейма **:fr**, пропущене через фільтр, який знаходиться в аспекті **$filter** слота **:prop** фрейма-заповнювача ролі **:akt** фрейма **:fr**

```
(passertq !! (:fr :akt :prop :value)
 ((setq :value (fget (setq :akt
(fgete :fr :akt)) :prop $value '(E)))
    (fexec :akt :prop '$filter))
 ((setq :value (fget 'ФІЗ.ОБ’ЄКТ :prop
$default '(E)))
    (fexec ' ФІЗ.ОБ’ЄКТ :prop '$filter))
)
```

Наступна лекція буде присвячена системі подання знань на логічній мові Prolog.