ФУНКЦІОНАЛЬНЕ ПРОГРАМУВАННЯ

Лекція 7.
СКЛАДНІ СТРУКТУРИ В ФУНКЦІОНАЛЬНИХ МОВАХ ПРОГРАМУВАННЯ

Язык **MFRL/PC** (Frame Representation Language with Matching for PC) основан на базе ФРЛ (FRL - Frame Representation Language, Roberts & Goldstein, MIT 1977) в МЭИ на кафедре прикладной математики в группе системного программирования под руководством Семеновой Е.Т.

Язык фреймов – технология, которая используется для представления знаний в области искусственного интеллекта. Фреймы сохраняются как онтологий множеств и подмножеств понятий кадров. Они похожи на иерархии классов в объектно-ориентированных языках, хотя их фундаментальные цели разработки отличаются. Фреймы сосредоточены на явном и интуитивно понятном представлении знаний, в то время как объекты базируются на инкапсуляции и сокрытия информации. Фреймы возникли при исследовании искусственного интеллекта и объектов, прежде всего, при разработке программного обеспечения. Тем не менее, на практике методы и возможности фреймовых и объектно-ориентированных языков значительно перекрываются.

Фрейм

Фрейм – это структура данных для представления стереотипной ситуации, например пребывание гостиной или сборы на вечеринку по случаю дня рождения ребенка. К каждому кадру прикрепляется несколько видов информации. Часть этой информации о том, как использовать фрейм. Часть о том, что можно ожидать дальше. Часть о том, что делать, если эти ожидания не подтвердились.

Мы можем думать о фрейме, как о сети узлов и отношений. "Топ-уровни" фрейма фиксированы, и представляют собой утверждения о предполагаемой ситуации, которые всегда верны. Нижние уровни имеют много терминальных "слотов", которые должны быть заполнены в зависимости от конкретных случаев или данных. Наборы фреймов связаны вместе в каркасные системы. Последствия важных событий отражаются преобразованиями между фреймами системы.

Синтаксис фрейма в расширенной нотации Бэкуса-Наура (метасимволы: "::=" "<" ">" "|" "{" "}" "[" "]") имеет следующий вид:

```
<фрейм> ::= ( <имя фрейма> { <слот> } )
<имя фрейма> ::= <атом>
<слот> ::= ( <имя слота> { <аспект> } )
<имя слота> ::= <отношение> | <свойство> | SELF
<отношение> ::= <атом>
<свойство> ::= <атом>
<аспект> ::= ( <имя аспекта> { <данное> } )
<имя аспекта> ::= <атом>
<данное> ::= ( <имя данного> { <комментарий> } ) | <функтор>
<функтор> ::= OR | ALT | NOT | IF | THEN | ELSE | FI
<имя данного> ::= <значение>
<значение> ::= <атом> | <d-пара> |
                <имя ЛИСП- или ФРЛ-функции> |
                <вызов ЛИСП- или ФРЛ-функции> |
                <имя фрейма> | <фрейм>
<комментарий> ::= ( <имя комментария> { <сообщение> } )
<имя комментария> ::= <метка>
<метка> ::= <атом>
<сообщение> ::= <s-выражение>
```

Типы данных MFRL/PC.

В MFRL/PC существуют три типа значений (данных): косвенное, вычислимое и прямое.

Косвенное значение.

Косвенное данное имеет комментарий (STATUS: INDIRECT) и может также содержать комментарии с метками SLOT: и FACET. Пример. Пусть имеются фреймы F1 и F2:

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

Тогда в ответ на запрос на данное из аспекта A слота S фрейма F1 будет выдано значение V2, т.к. значение F2 является косвенным.

Вычислимое значение.

Вычислимое данное - это такое данное, которое содержится в аспектах $IF-ADDED, $IF-REMOVED, $IF-NEEDED, $IF-INSTANTIATED, $REQUIRE, $IF-GET, $IF-PUT, $IF-REM и не имеет комментария (STATUS: NOEVAL) либо содержится в аспектах $VALUE, $DEFAULT и имеет комментарий (STATUS: EVAL). Пример. Пусть имеется фрейм F

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

Тогда в ответ на запрос данных из аспекта $VALUE слота S фрейма F будет выдано не данное с именем LIST, а данное с именем (A B C), т.е. с именем, полученным в результате вычисления функции LIST с аргументами A, B и C. То же самое получится и в случае, когда фрейм F имеет вид

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

=== Прямое значение. ===

Все остальные данные являются прямыми. Они извлекаются из сети фреймов без какой-либо дополнительной обработки.

Типы комментариев

В ФРЛ имеются несколько типов комментариев:

- комментарий с меткой STATUS: используется для индикации метода обработки данного при его извлечении. Возможными сообщениями для этого комментария являются EVAL, NOEVAL и INDIRECT;
- комментарии с метками SLOT: и FACET: используются для указания референта косвенного данного. Они используются совместно с комментарием (STATUS: INDIRECT);
- комментарии с метками PARM: и PARMQ: используются при задании аргументов для присоединенных процедур;
- комментарий с меткой FINHERIT: используется для локального управления наследованием. Если сообщение есть CONTINUE, то содержащий его слот наследует данные из своих AKO-прототипов даже в случае, если он сам и содержит искомые данные. Сообщение STOP запрещает наследование данных, искомых в содержащем его слоте;

- комментарий (TYPE: <тип>) обеспечивает возможность избирательного вызова процедур функциями FPROC, FNEED и FEXEC;
- комментарий с меткой IN: вставляется системой в данные при их извлечении для идентификации фрейма, слота и аспекта, из которых они были извлечены.

Трассировка

В ФРЛ имеется возможность трассировки, т.е. выполнения определенных программ пользователя, при внесении изменений в любой фрейм, слот, аспект, данное, комментарий и сообщение. Для этой цели в системе имеются специальные средства, через которые осуществляется воздействие пользователя на работу ФРЛпроцессора. К таким средствам относится процедура FTRACE. С помощью FTRACE пользователь может подключать к системе процедуры (будем называть их трассирующими), которые управляют запуском присоединенных процедур. Трассирующие процедуры могут быть присоединены к любому фрейму с заданной структурой. Для подключения этих процедур к работе ФРЛ-процессора необходимо выполнить функцию: (FTRACE <имя-фрейма> <список-пар>), где <имя-фрейма> - имя фрейма, содержащего трассирующие процедуры, а <список-пар> - список вида ((<мета-имя-подструктуры><условие> ...).

<Мета-имя-подструктуры> - это есть FRAME, SLOT, VALUE, LABEL или MESSAGE.

<Условие> представляет один из следующих атомов: IF-ADDED, IF-REMOVED, IF-GETED, что интерпретируется следующим образом:

- IF-ADDED - трассирующие процедуры запускаются при добавлении в систему структуры с соответствующим мета-именем;
- IF-REMOVED - трассирующие процедуры запускаются при удалении из системы структуры с соответствующим мета-именем;
- IF-GETED - трассирующие процедуры запускаются при извлечении из сети структуры с соответствующим мета-именем.

Можно использовать и такое обращение: (TRACE <имя фрейма>). В этом случае будут об'явлены трассирующими все процедуры из фрейма <имя фрейма>.

Для отключения режима трассировки следует использовать функцию FUNTRACE. Синтаксис: (FUNTRACE). Ниже приводится структура фрейма, используемого процедурой FTRACE.

```
(<имя фрейма>
   (FRAME ($IF-ADDEED <процедуры, выполняемые при
                       занесении нового фрейма в систему>)
          ($IF-REMOVED <процедуры,выполняемые при удалении
                        фрейма из системы>)
          ($IF-GETED <процедуры, извлекающие информацию из
                      фрейма>)
   )
   (SLOT ($IF-ADDED <процедуры, выполняемые при занесении
                     нового слота в какой-нибудь фрейм>)
         ($IF-REMOVED <процедуры, выполняемые при удалении
                       слота из какого-либо фрейма>)
         ($IF-GETED <процедуры, извлекающие информацию из
                     слота>)
   )
   (FACET ($IF-ADDED <процедуры, выполняемые при занесении
                      нового аспекта в какой-либо слот>)
          ($IF-REMOVED <процедуры, выполняемые при удалении
                        аспекта из какого-либо слота>)
          ($IF-GETED <процедуры, извлекающие значение из
                      аспекта>)
   )
```

```
(VALUE ($IF-ADDED <процедуры, выполняемые при занесении
                   нового значения в какой-либо аспект>)
       ($IF-REMOVED <процедуры, выполняемые при удалении
                     значения из какого-либо аспекта>)
       ($IF-GETED <процедуры, извлекающие значение из
                   данного>)
  )
  (LABEL ($IF-ADEED <процедуры, выполняемые при занесении
                     нового комментария в какое-либо данное>)
         ($IF-REMOVED <процедуры, выполняемые при удалении
                       комментария из какого-либо данного>)
         ($IF-GETED <процедуры, извлекающие комментарий из
                     какого-либо данного>)
  )
  (MESSAGE ($IF-ADDED <процедуры, выполняемые при
                       занесении нового сообщения в какой-либо
                       комментарий>)
           ($IF-REMOVED <процедуры, выполняемые при
                         удалении сообщения из какого-либо
                         комментария>)
  )
)
```

Примечание: Все компоненты указанной фрейм-структуры носят факультативный характер.

Краткое описание функций языка MFRL/PC

Обозначения для функций MFRL/PC

В таблицах, приводимых ниже, будем использовать следующие обозначения для аргументов функций в языке MFRL/PC:

- f - имя фрейма;
- fs - фрейм-структура;
- s - имя слота;
- ss - слот;
- a - имя аспекта;
- as - аспект;
- v - имя данного;
- vs - данное;
- l - имя комментария (метка);
- ls - комментарий;

- m - имя сообщения;
- ms - сообщение;
- pnl - список имен процедур (procedure name list)
- fnl - список имен фреймов (frame name list)
- fvl - список данных ФРЛ (frame value list)
- fn - имя функции (function name)

Примечание: Факультативные аргументы ФРЛ-функций следуют за обязательными и в данном описании отделяются двоеточием. Альтернативные аргументы указаны в фигурных скобках.

Константы MFRL/PC

Синтаксис	Семантика
AKO-INSTANCE	Список свойств для хранения имен инверсных отношений (CDR '*AKO-INSTANCE*) => ((AKO . INSTANCE) (INSTANCE . AKO))
AKO	AKO
INSTANCE	INSTANCE
$VALUE	$VALUE
$DEFAULT	$DEFAULT
$IF-NEEDED	$IF-NEEDED
$IF-REMOVED	$IF-REMOVED
$IF-ADDED	$IF-ADDED
$IF-INSTANTIATED	$IF-INSTANTIATED
$REQUIRE	$REQUIRE
$IF-GETED	$IF-GETED
$IF-ADD	$IF-ADD

$IF-REM $IF-REM
$IF-GET $IF-GET

Стеки MFRL/PC

Имя стека	Назначение	Используется функциями
FRAMES (список свойств)	Содержит имена активных фреймов	DEFRAME, FRESET, FPRINT, FASSERT, FSAVE, DEFRAMEQ, FASSERTQ, FDESTROY
PROCEDURES (список свойств)	Содержит имена активных процедур	PASSERT, PRESET, PSAVE, PASSERTQ
FGENAMELIST (переменная)	Содержит информацию для генерации уникальных имён фреймов (реализован как список свойств)	FGENAME, FGETNAME

=== Глобальные переменные MFRL/PC ===

Синтаксис	Семантика
FRAME	Тело фрейма сразу после FRAME? или FNAME?
FNAME	Имя фрейма сразу после FRAME? или FNAME?
DESCR	Тело дескриптора сразу после DNAME?
DNAME	Имя дескриптора сразу после DNAME?
MATCHED	Список сопоставившихся объектов после FMATCH?
UNMATCHED	Список несопоставившихся объектов после FMATCH?

=== Изменение сети фреймов и процедур ===

Синтаксис	Семантика
(PASSERT fn bd)	Определяется или переопределяется процедура с именем fn и телом bd. Имя процедуры заносится в стек *PROCEDURES*. Результат - имя процедуры.
(PASSERTQ fn bd)	То же что и PASSERT, но при обращении аргументы не вычисляются.
(DEFRAME f : ss1 ... ssn)	Создается новый фрейм, содержащий указанные слоты. Результат - имя созданного фрейма. Присоединенные процедуры не активизируются.
(DEFRAMEQ f : ss1 ... ssn)	Аналог DEFRAME, но аргументы DEFRAMEQ не вычисляются при обращении.
(FASSERT f : ss1 ... ssn)	Создается новый или пополняется старый фрейм. Результат - имя фрейма. Присоединенные процедуры выполняются.
(FASSERTQ f : ss1 ... ssn)	Аналог FASSERTQ, но аргументы FASSERTQ

	не вычисляются при обращении.
(FRENAME f1 : f2)	Фрейм с именем f1 переименовывается во фрейм с именем f2. Результат - новое имя.
(FNAME {f fs})	Результат - имя фрейма. Если такой фрейм не найден - он создается.
(FRAME {f fs})	Результат - указатель на фрейм-структуру. Если фрейм не найден - он создается
(FPUT- f : s a v l m)	В тело фрейма f, добавляется слот s, аспект a, данное v, метка l, комментарий m. Если какая либо из структур: f,s,a, v,l,m уже существовала, то в нее либо добавляется новая информация, либо эта структура остается без изменения. Если к моменту выполнения FPUT какая либо из структур: f,s,a,v,l,m не существовала - она создается. Присоединенные процедуры не выполняются
(FPUT-STRUCTURE- f) (FPUT-STRUCTURE- f ss) (FPUT-	То же, что и в FPUT-, но последний аргумент обращения трактуется как готовая

STRUCTURE- f s as) (FPUT-STRUCTURE- f s a vs) (FPUT-STRUCTURE- f s a v ls) (FPUT-STRUCTURE- f s a v l ms)	соответствующая структура, добавляемая целиком во фрейм
(FPUT f : s a v l m)	В тело фрейма f, добавляется слот s, аспект a, данное v, метка l, комментарий m. Если какая либо из структур: f,s,a, v,l,m уже существовала, то в нее либо добавляется новая информация, либо эта структура остается без изменения. Если к моменту выполнения FPUT какая либо из структур: f,s,a,v,l,m не существовала - она создается. Присоединенные процедуры выполняются
(FPUTV f s v)	=(FPUT f s $VALUE v)
(FPUT-STRUCTURE f) (FPUT-STRUCTURE f ss) (FPUT-STRUCTURE f s as) (FPUT-STRUCTURE f s a vs) (FPUT-STRUCTURE f s a v ls) (FPUT-	То же, что и в FPUT, но последний аргумент обращения трактуется как готовая соответствующая структура, добавляемая целиком во фрейм

STRUCTURE f s a v l ms)	
(FPUT-STRUC f sb)	=(FPUT-STRUCTURE f sb)
(FPUT-STRUC f s ab)	=(FPUT-STRUCTURE f s ab)
(FPUT-STRUC f s a d)	=(FPUT-STRUCTURE f s a d)
(FPUT-STRUC f s a v c)	=(FPUT-STRUCTURE f s a v c)
(FPUT-STRUC f s a v l m)	=(FPUT-STRUCTURE f s a v l m)
(FDELETE f : s a v l m)	То же, что и FREMOVE, но присоединенные процедуры не активизируются и результат - указатель на измененную подструктуру фрейма.
(FDEL-STRUCTURE f) (FDEL-STRUCTURE f ss) (FDEL-STRUCTURE f s as) (FDEL-STRUCTURE f s a vs) (FDEL-STRUCTURE f s a v ls) (FDEL-STRUCTURE f s a v l ms)	Удаляет подструктуру фрейма f, локализуемую остальными аргументами обращения. Результат - измененная подструктура или NIL, если удаления не произошло. Присоединенные процедуры не выполняются. Число аргументов - от 1 до 6
(FREMOVE f : s a v l m)	Удаляет из фрейма подструктуру, локализуемую аргументами обращения.

	Последний аргумент обращения задает имя удаляемой подструктуры. Число аргументов - от 1 до 6. Результат - последний аргумент в обращении. Присоединенные процедуры выполняются.
(FREM-STRUCTURE f) (FREM-STRUCTURE f ss) (FREM-STRUCTURE f s as) (FREM-STRUCTURE f s a vs) (FREM-STRUCTURE f s a v ls) (FREM-STRUCTURE f s a v l ms)	То же что и FDEL-STRUCTURE, с той разницей что присоединенные процедуры не выполняются
(FREM-STRUC f s a v l m)	=(FREM-STRUCTURE f s a v l m)
(FREP-STRUCTURE f) (FREP-STRUCTURE f ss) (FREP-STRUCTURE f s as) (FREP-STRUCTURE f s a vs) (FREP-STRUCTURE f s a v ls) (FREP-STRUCTURE f s a v l ms)	Последний аргумент обращения трактуется как структура соответствующего уровня. Она полностью заменяет во входном фрейме соответствующую структуру. Результат - измененная подструктура. Число аргументов - от 1 до 5. Присоединенные процедуры выполняются.

(FREP-STRUCTURE- f) (FREP-STRUCTURE- f ss) (FREP-STRUCTURE- f s as) (FREP-STRUCTURE- f s a vs) (FREP-STRUCTURE- f s a v ls) (FREP-STRUCTURE- f s a v l ms)	Последний аргумент обращения трактуется как структура соответствующего уровня. Она полностью заменяет во входном фрейме соответствующую структуру. Результат - измененная подструктура. Число аргументов - от 1 до 5. Присоединенные процедуры не выполняются.
(FREP-STRUC f s a v l m)	=(FREP-STRUCTURE f s a v l m)
(FINSTANTIATE f1 : f2)	Создается экземпляр фрейма f1 с именем f2. Если f2 опущено, то создается фрейм с системным именем, в создаваемый экземпляр помещаются также значения, вырабатываемые процедурами, присоединенными к аспектам $IF-INSTANTIATED каждого слота фрейма f. Результат - имя созданного фрейма.
(FCLEAN f s : a)	Удаляет из фрейма f слота s аспекта a все данные, которые не удовлетворяют процедурам, наследуемым из аспекта $REQUIRE. Если аспект a опущен, то

	a=$VALUE. Возвращает список удаленных данных.
(FINSTANCE f sb*)	Создает экземпляр фрейма f со слотами s1 s2 ... sN.
(FREVADD s)	Создает обратную ссылку s из фрейма :VALUE на фрейм :FRAME (в аспекте :FACET).
(FREVREM s)	Удаляет обратную ссылку s из фрейма :VALUE на фрейм :FRAME (в аспекте :FACET).
(FORWADD s)	Создает прямую ссылку s из фрейма :VALUE на фрейм :FRAME (в аспекте:FACET).
(FORWREM s)	Удаляет прямую ссылку s из фрейма :VALUE на фрейм :FRAME (в аспекте:FACET).
(PRESET : pnl)	Указанные процедуры удаляются из системы. Результат список имен удаленных процедур. Если pnl в обращении опущен, то считается, что pnl=*PROCEDURES* .
(FRESET : fnl)	Указанные фреймы затираются в оперативной памяти. Присоединенные процедуры не

	выполняются. Результат - список имен затертых фреймов. Если fnl в обращении опущен, то считается, что fnl=*FRAMES* .
(DEDESCR q sb*)	Создает новый дескриптор q со слотами s1 ... sN.
(DINSTANCE q sb*)	Создает экземпляр дескриптора q со слотами s1 ... sN.

Литература

1. Frame language
2. A Framework for Representing Knowledge, Marvin Minsky, MIT-AI Laboratory Memo 306, June, 1974.
3. Методические указания по курсу «Основы программирования» Языки и системы представления знаний (язык программирования ФРЛ), Байдун В.В., Бунин А.И., Чернов П.Л. ── М.: Моск. энерг. ин-т, 1993. ── 44 с.