Зинкина Н.С. Методы спецификации динамических структур данных. // Проблемы информатики в образовании, управлении, экономике и технике: Сб. статей IX Междунар. научно-техн. конф. – Пенза: ПДЗ, 2009. – С. 57-60.

МЕТОДЫ СПЕЦИФИКАЦИИ ДИНАМИЧЕСКИХ СТРУКТУР ДАННЫХ

Н.С. Зинкина

Пензенский государственный университет,
г. Пенза, Россия

Предлагаются методы спецификации динамических структур данных для сетей абстрактных машин.

Zinkina N.S. Methods of specification of dynamic data structures.

Methods of specification of dynamic data structures for networks of abstract machines are suggested.

В сетях абстрактных машин можно использовать активные элементы FS-пространства, или динамические структуры данных – «очереди» и «стеки». Для реализации FS-пространства целесообразно использовать ассоциативную (или частично-ассоциативную) память, с помощью которой возможна эффективная реализация квантифицированных операторов выбора $\tilde{\exists}!$ (выбрать произвольный единственный элемент из элементов, удовлетворяющих некоторому условию) и $\tilde{\forall}$ (выбрать все элементы, удовлетворяющие некоторому условию).

Обозначим именем FIFO («первый на входе – первый на выходе») объект, или динамическую структуру данных типа «очередь с проталкиванием», а именем LIFO («последний на входе – первый на выходе») – объект типа «стек», или «магазинная память». Введение таких динамических структур данных, как очереди и стеки, в FS-пространство позволяет также реализовать сетевую память для кэширования данных и эффективной организации связей между процессами.

Пусть $FIFO_i(x, y)$ – динамическая структура данных типа «очередь», элементами которой являются кортежи, содержащие значения предметных переменных x и y различных типов; одноименная предикатная константа, или бинарный предикатный символ $FIFO_i$ будет использован далее в логико-алгебраических выражениях. Введем квантифицированный оператор $\vec{\exists}!$, позволяющий выбрать первый элемент очереди. Тогда выражение m_{out}, использующее данный оператор, можно записать следующим образом:

$$m_{out} = [f_{NE}(FIFO_i)]([\vec{\exists}!(x, y)FIFO_i(x, y)](\{FIFO_i(x, y) \leftarrow \text{false},$$

$$f_{buff}(x, y) \leftarrow \text{true}\} \vee R^{E}) \vee R^{E}),$$

где f_{NE} – высказывательная функция (унарный предикат), принимающая значение true в том случае, когда очередь не пуста, т.е. структура данных $FIFO_i$ содержит по крайней мере один кортеж; f_{buff} – «буферная» высказывательная функция (бинарный предикат), в область истинности которой включаются выбираемые из очереди $FIFO_i$ кортежи; оператор $\vec{\exists}!(x, y)\, FIFO_i(x, y)$ выбирает из упомянутой совокупности кортежей первый кортеж; при обновлении $FIFO_i(x, y) \leftarrow$ false выбранный кортеж удаляется из области истинности бинарного предиката $FIFO_i$;

обновление $f_{buff}(x, y) \leftarrow$ true добавляет к области истинности бинарного предиката f_{buff} новый кортеж. Отличительной особенностью использования объекта FIFO является то, что после удаления из области истинности первого кортежа головным элементом очереди станет следующий кортеж. Запись элемента в очередь описывается, например, следующим выражением:

$$m_{in} = [f_{NF}(FIFO_i)]([(\tilde{\exists}!x)f_1(x)]([(\tilde{\exists}!x)f_2(y)]({FIFO_i(x, y) \overset{\circ}{\longleftarrow} true} \vee$$

$$\vee R^E) \vee R^E) \vee R^E),$$

где f_{NF} – унарный предикат, принимающий значение true в том случае, когда в очереди есть по крайней мере одно свободное место для размещения нового кортежа; операторы $(\tilde{\exists}!x)f_1(x)$ и $(\tilde{\exists}!x)f_2(y)$ выбирают по одному значению переменных x и y из области истинности унарных предикатов $f_1(x)$ и $f_2(y)$; оператор обновления $FIFO_i(x, y) \overset{\circ}{\longleftarrow} true$ помещает новый кортеж в очередь, причем кружок над стрелкой указывает на то, что новый кортеж становится последним элементов в очереди.

Квантифицированный оператор $\tilde{\forall}$ может быть использован для удаления всех элементов из очереди $FIFO_i$:

$$m_{out\_all} = [f_{NE}(FIFO_i)]([\tilde{\forall}(x, y)FIFO_i(x, y)]({FIFO_i(x, y) \overset{*}{\longleftarrow} false,$$

$$f_{buff}(x, y) \overset{*}{\longleftarrow} true\} \vee R^E) \vee R^E).$$

Здесь оператор $\tilde{\forall}(x, y)FIFO_i(x, y)$ выбирает все пары вида <x, y>, входящие в область истинности структуры $FIFO_i(x, y)$. В данном случае в блоке модуля будет выполнено несколько обновлений $FIFO_i(x, y) \overset{*}{\longleftarrow} false$ для каждого выбранного кортежа, т.е. все кортежи исключаются из области истинности структуры $FIFO_i(x, y)$ и включаются в область истинности бинарного предиката f_{buff}. Звездочка над стрелкой означает, что операции обновления выполняются для всех выбранных оператором $\tilde{\forall}$ кортежей.

Модуль m'_{out} удаляет кортеж из стека $LIFO_i$:

$$m'_{out} = [f_{NE}(LIFO_i)]([\vec{\exists}!(x, y)LIFO_i(x, y)]({LIFO_i(x, y) \leftarrow false,$$

$$f_{buff}(x, y) \leftarrow true\} \vee R^E) \vee R^E).$$

Выражение для модуля m'_{out} выполняется аналогично выражению для модуля m_{out} за исключением того, что оператор $\vec{\exists}!$, примененный к объекту типа LIFO, выбирает в структуре $LIFO_i$ кортеж, находящийся в вершине стека. Запись элемента в стек выполняет, например, следующий модуль:

$$m'_{in} = [f_{NF}(LIFO_i)]([(\tilde{\exists}!x)f_1(x)]([(\tilde{\exists}!y)f_2(y)]({LIFO_i(x, y) \overset{\bullet}{\longleftarrow} true} \vee$$

$$\vee R^E) \vee R^E) \vee R^E).$$

Здесь точка над стрелкой означает, что новый кортеж заносится в вершину стека. Аналогично выражению для модуля m_{out_all} выполняется и операция удаления всех элементов из стека, необходимо в соответствующем выражении заменить все вхождения имени FIFO на LIFO.

Состояние любой очереди или стека – пусты, непусты, заполнены полностью, не заполнены полностью – в любой момент времени позволяют проверить соответствующие унарные предикаты f_E, f_{NE}, f_F, f_{NF}, а число элементов, находящихся в очереди или стеке, может быть определено с помощью унарной функции f_{number}, значение которой можно проверять в любых модулях имитационной модели.

В общем случае возможно возникновение и более сложных ситуаций при распределении ресурсов в сетях хранения и обработки данных. Например, ресурс может состоять из многих физических узлов и иметь несколько единиц, за разделяемое или монопольное использование которых конкурируют многие процессы; узлы сети, реализующие единицы абстрактного ресурса, могут использоваться другими процессами. Возникает проблема выбора стратегий ожидания или захвата единиц ресурса, проблема разрешения тупиковых ситуаций.

Библиографический список

1. Зинкин С.А. Сети абстрактных машин высших порядков в проектировании систем и сетей хранения и обработки данных (базовый формализм и его расширения) // Известия высших учебных заведений. Поволжский регион. Технические науки. – 2007. – № 3. – С. 13 – 22.

2. Зинкин С.А. Сети абстрактных машин высших порядков в проектировании систем и сетей хранения и обработки данных (механизмы интерпретации и варианты использования) // Известия высших учебных заведений. Поволжский регион. Технические науки. – 2007. – № 4. – С. 37 –51.