МОДЕЛИРОВАНИЕ ОДНОРОДНЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ

С.В. Панков

Рассматривается класс асинхронных однородных вычислительных систем (ОВС), использующих произвольное (сколь угодно большое) число однотипных вычислительных устройств с регулярной топологией связей между ними. При этом, ОВС являются, либо специализированными системами с фиксированной структурой, либо универсальными с программируемой структурой, но уже настроенными на решение определённой задачи. Проблема обоснования функциональной корректности (правильности работы с точки зрения вычисляемой функции) для таких систем является актуальной.

Осуществляется моделирование ОВС на основе *L*-программ, логико-математической модели асинхронных параллельных вычислений, предложенной впервые в [1]. Построенная модель ориентированна на применение логических методов верификации *L*-программ [2] для анализа корректности ОВС. Подобный подход применялся в [3] к анализу поведения клеточных автоматов. Благодаря однотипности вычислительных процессов, регулярности структуры связей в ОВС, модель (как и в [3]) не зависит от числа процессов, что обеспечивает также и независимость анализа модели от их числа.

1. Описание ОВС.

ОВС определяется в виде, ориентированном на моделирование *L*-программой, как система $S=<P,C>$, где P – множество процессов, а C – множество линий связи. Множество P разбивается на три подмножества $P=P_{in}\cup P_{calc}\cup P_{out}$. Здесь P_{in} – множество процессов *ввода*, имеющих по одному выходу (называемому *входом системы*) и ни одного входа. P_{calc} – это множество однотипных *вычислительных* процессов, имеющих конечное ненулевое число входов – n, выходов – m, и вычисляющих набор функций $<f_1,...,f_m>$. Входные данные процесса снимаются со входов, вычисляются функции на этом наборе данных, а затем результаты помещаются на сотвествующие выходы. (Предполагается, что все функции определены на входном наборе данных.) P_{out} – это множество процессов *вывода*, имеющих по одному входу (называемому *выходом системы*) и ни одного выхода. Выходы процессов из P_{calc} соединены с входами соседних процессов этого же множества линиями связи так, что образуется регулярная структура. (Имеется ввиду структура графа, вершины которого соответствуют процессам из P_{calc}, а ребра – линиям связи между ними). На C также накладываются следующие ограничения: 1) Каждый выход процесса из P связан точно с одним входом некоторого процесса. 2) Каждый вход процесса из P связан только с одним выходом некоторого процесса. ОВС также характерризуется потоками данных на её входах и выходах.

Процесс p из P_{calc} может находиться в одном из двух состояний: состоянии *ожидания* (в нём все выходы p свободны), или состоянии *выполнения*. Процесс p может перейти из состояния ожидания в состояние выполнения, если справедливо следующее условие *готовности к выполнению*: все входы p содержат данные. Само выполнение считается одномоментным и сопровождается снятием данных со всех входов, заполнением результатами вычисления всех выходов p. Передача данного с выхода процесса p на вход приемника p' одномоментна и может произойти, если выполняется следующее условие *готовности к передаче*: выход процесса p заполнен, вход процесса p' свободен, а сам процесс p' (если p' не совпадает с p) находится в состоянии ожидания.

Состояние системы S определяется состоянием потоков данных на её входах и выходах, наличием данных на входах и выходах процессов, а также самими данными. В начальном состоянии все входы и выходы системы не содержат данных, все процессы-вычислители находятся в состоянии ожидания, входные потоки данных системы непусты, а выходные – пусты.

Шаг работы системы S состоит в следующем: 1) происходит заполнение данными произвольного множества входов системы с непустыми входными потоками; 2) произвольное множество готовых к выполнению процессов переходят из состояния ожидания в состояние выполнения; 3) осуществляется передача данных с произвольного множества готовых к передаче выходов процессов; 4) происходит снятие данных с произвольного множества заполненных выходов системы. (Объединение перечисленных множеств должно быть непустым.) Переход из состояния выполнения в состояние ожидания вычислительным процессом осуществляется после передачи всех результатов процессам-приемникам. Пусть Q – это класс всех состояний системы S. Запись $q\rightarrow q'$ будет означать *достижимость* q' из q за один шаг. Состояние называется *заключительным*, если в нём система не может сделать ни одного шага. Пусть $I,E\subseteq Q$. Система S *корректна* относительно предусловия I и постусловия E ($[I]\ S\ [E]$), если любое заключительное состояние, достижимое из I, принадлежит E.

2. Моделирование ОВС.

Для моделирования L-программой работы системы S, сначала определяется язык предметной области – L, как язык логики предикатов первого порядка.

Сорта: *ПР* – процессы из P;

ВХ – номера входов процессов $\{1, 2,...,m\}$;

ВЫХ – номера выходов процессов $\{1, 2,...,n\}$;

ДАН – данные;

НАТ – натуральные числа и 0.

Функции: $+$: $НАТ \times НАТ \rightarrow НАТ$ – сложение;

кол: $ПР \rightarrow НАТ$ – задает количество срабатываний процесса, с момента начала работы системы (в начальный момент $кол(p)=0$ для всех $p \in P$);

$данное_{in}$: $ВХ \times ПР \times НАТ \rightarrow ДАН$ – $данное_{in}(k,p,i)$ задает данное, содержащееся на k-том входе процесса p перед (i+1)-вым срабатыванием p (третий аргумент введён для удобства анализа модели, так выходной поток данных системы на текущий момент для $p \in P_{out}$ представляется набором значений $данное_{in}(1,p,кол(p))$,,...,$данное_{in}(1,p,1)$);

$данное_{out}$: $ПР \times ВЫХ \times НАТ \rightarrow ДАН$ – $данное_{out}(p,k,i)$ задает данное, содержащееся на k-том вхыоде процесса $p \in P_{calk}$ после i-го перехода p из состояния ожидания в состояние выполнения (также определяет остаток входного потока данных системы на текущий момент для $p \in P_{in}$ набором значений $данное_{out}(p,1,кол(p)+1)$, $данное_{out}(p,1,кол(p)+2)$,...);

Предикаты: *связь*: $ПР \times ВЫХ \times ВХ \times ПР$ – задает линии связи;

ввод: *ПР* – задаёт процессы ввода;

вычисление: *ПР* – задаёт вычислительные процессы;

вывод: *ПР* – задаёт процессы вывода;

$заполнен_{in}$: $ВХ \times ПР$ – задаёт наличие данных на входах процессов;

$заполнен_{ou}$: $ПР \times ВЫХ$ – задаёт наличие данных на выходах процессов.

Переменные: $пр,пр'$:*ПР*; $вх$, $вх'$:*ВХ*; $вых$, $вых'$:*ВЫХ*; $y,x_1,...,x_n$:*ДАН*; i,i':*НАТ*.

Язык L также содержит обычные арифметические функции и отношения. Знаками $\vee$, $\wedge$, 爾, ®, " и $ обозначают обычные логические связки и кванторы. Далее под Q понимается класс всех L-структур (представляющих состояния системы S). Модель в виде L-программы действует по шагам, преобразуя одну L-структуру в другую. Произвольный шаг моделирующей L-программы будет реализовывать шаг работы системы S. Согласно своему определению [2], L-программа задается конечной совокупностью правил вида условие ▪действие (где условие – произвольная L-формулы, а действие – L-формула специального вида [2]). Эти L-формулы зависят от набора свободных переменных. Правило может исполниться одновременно для нескольких (выбираемых недетерминированно) наборов значений свабодных переменных, для которых истинно условие. Кроме того, одновременно могут сработать несколько (также выбираемых недетерминированно) правил. В L-программе может использоваться средство синхронизации $ работы, как в рамка одного правила, так и на уровне нескольких правил [2].

Введём следующие обозначения формул:

$READY_{load}(пр)$: $\exists y(данное_{out}(пр,1,кол(p)+1)=y)$ – выражает существование данных в остатке входного потока для процесса ввода *пр*;

$READY_{calc}(пр)$: $\forall вх'(заполнен_{in}(вх',пр))$ – выражает условие готовности вычислительного процесса к выполнению;

$READY_{send}(пр,вых,вх',пр')$: $связь(пр,вых,вх',пр') \wedge заполнен_{out}(пр,вых) \wedge \neg заполнен_{in}(вх',пр') \wedge$

$(пр' \neq пр \rightarrow \forall вых'(\neg заполнен_{out}(пр',вых')))$ – ыражает условие готовности к передаче данного с выхода *вых* процесса *пр* на вход *вх*' процесса *пр*'.

Моделирующая L-программа состоит из четырех правил, моделирующих соответственно: 1) загрузку данных на входы системы; 2) переход вычислительных процессов из состояния ожидания в состояние выполнения (обратный переход происходит автоматически, после передачи всех результатов приемникам); 3) передачу данных приемникам; 4) снятие данных с выходов системы.

L-программа:

1) $ввод(пр) \wedge READY_{load}(пр) \wedge \neg заполнен_{out}(пр,1) \wedge i=кол(пр)+1 \Rightarrow$

$заполнен_{out}(пр,1) \wedge кол(пр)=i$

2) **$** *вх*,*вых*,*y*

$вычисление(пр) \wedge READY_{calc}(пр) \wedge$

x_1=*данное*$_{in}$(1,*пр*,*кол*(*пр*))∧…∧x_n=*данное*$_{in}$(1,*пр*,*кол*(*пр*))∧

($y=f_1(x_1,...,x_n)$∨…∨$y=f_m(x_1,...,x_n)$)∧ *i*=*кол*(*пр*)+1 ⇒

¬*заполнен*$_{in}$(*вх*,*пр*)∧*заполнен*$_{out}$(*пр*,*вых*)∧*данное*$_{out}$(*вых*,*пр*,*i*)=*y*∧*кол(пр)*=*i*

3) READY$_{send}$(*пр*,*вых*,*вх*',*пр*')∧*y*=*данное*$_{out}$(*пр*,*вых*,*кол*(*пр*))∧*i*'=*кол*(*пр*') ⇒

¬*заполнен*$_{out}$(*пр*,*вых*)∧*заполнен*$_{in}$ (*вх*',*пр*')∧*данное*$_{in}$(*вх*',*пр*',*i*')=*y*

4) *вывод*(*пр*)∧*заполнен*$_{in}$ (1,*пр*)∧*i*=*кол*(*пр*)+1 ⇒ ¬*заполнен*$_{in}$ (1,*пр*)∧*кол*(*пр*)=*i*

Первое правило может исполниться одновременно для нескольких процессов ввода (значений переменной *пр*) с непустыми входными потоками и незаполнеными выходами. При этом, значение переменной *i* подбирается так, чтобы увеличить с её помощью количество срабатываний процесса на 1. В результате исполнения действия правила заполняются выходы процессов, и изменяется количество срабатываний.

Второе правило может исполниться одновременно для нескольких вычислительных процессов, готовых к выполнению. Переменные $x_1,...,x_n$ используются для упрощения записи условия правила. Запись вида $f_k(x_1,...,x_n)$ – это *L*-терм, вычисляющий соответствующую функцию, где $1 \le k \le n$. (Вместо формулы вида $y=f_k(x_1,...,x_n)$ может использоваться *L*-формула, определяющая значение *y* функции f_k.) Синхронизатор **$** *вх*,*вых*,*y* обеспечивает одновременное освобождение всех входов, вычисление значений всех функций и заполнение этими значениями всех соответствующих выходов процесса. Третье и четвёртое правило поясняются аналогично.

ЛИТЕРАТУРА:

1. С.П. Крицкий "Модель асинхронных вычислений в структурах и языки программирования" // Методы трансляции. Ростов-на-Дону. 1981. с. 92-100
2. С.П. Крицкий, С.В. Панков "О верификации асинхронных программ продукционного типа"// Программирование. 1994. № 5. с. 40-52
3. С.В. Панков Материалы Всероссийской научной конференции "Научный сервис в сети ИНТЕРНЕТ", Новороссийск, 19-24 сентября 2007 г., Изд-во Московского Университета, с. 85 – 88