УДК 681.3.06

# ОТЛАДКА ФУНКЦИОНАЛЬНО-ПОТОКОВЫХ ПАРАЛЛЕЛЬНЫХ ПРОГРАММ С ПОДСТАНОВКОЙ ИНТЕРВАЛЬНЫХ ЗНАЧЕНИЙ

**Удалова Ю.В.**
**научный руководитель д-р техн. наук Легалов А.И.**
***Институт Космических и Информационных Технологий Сибирского Федерального Университета***

## 1. Введение

В статье предлагаются возможности отладки функционально-потоковых параллельных программ, обладающих рядом отличий от параллельных программ ориентированных на процессы или потоки (нити). Модель вычислений определяет функционально-потоковую параллельную программу как ациклический информационный граф, в котором вершины являются операторами, а дуги определяют связи между ними. Параллельное выполнение основано на том, что одновременно выполняться могут только те операторы, между которыми нет информационных связей. Чтобы оператор мог быть вычислен, необходимо выполнение всех связанных с ним предшественников.

Существующие отладчики функционально-потоковых параллельных программ предоставляют достаточно развитый инструментарий, но не настолько, как отладчики последовательных или многопроцессных программ. Например, часть из них обеспечивают визуализацию графов программы, предоставляют возможность выбора шага отладки, прямого и обратного хода отладки, поддерживают опции отладки частично не законченных программ.

Для отладки функционально-потоковой параллельной программы в статье предлагается режим проверки формул, поддерживающий расстановку логических условий (требований пользователя) на информационном графе программы и два способа определения входных данных программы для отладки: как точных значений, либо как их интервальных оценок.

Разработанный режим добавляет возможности оперирования дополнительными пользовательскими формулами и идентификаторами при отладке, тем самым, позволяя описать спецификацию к вычислениям программы или провести дополнительные вычисления, не изменяя основного кода программы. А возможность выполнения программы в режиме отладки над интервальными оценками входных значений предоставляет инструментарий для исследования корректности функционально-потоковой параллельной программы относительно спецификации пользователя над обобщенным множеством начальных данных.

## 2. Отладка в режиме проверки формул

Режим проверки формул позволяет пользователю закрепить за произвольными узлами-операторами информационного графа программы собственные утверждения (рис. 1), являющиеся выражениями на языке программирования, использующие в качестве значений аргумент функции или значения, вычисленные любым узлом-оператором графа. Если отлаживаемая функция не является рекурсивной, отладка выполняется за один шаг, на котором вычисляются все узлы графа и все дополнительные утверждения.

Вершины графа, содержащие пользовательские утверждения, помечаются как истинные, если все выражения, введенные пользователем, возвращают истину, или как ложные, если хотя бы одно из выражений не равно истине. Для каждой такой вершины вместе со значением соответствующего оператора программы можно увидеть вычисленные значения утверждений. Пользовательская формула может иметь не только логическое значение, но и любое другое: целочисленное, вещественное, строковое, но в

этом случае узел графа будет помечен как ложный. Если отлаживаемая функция является рекурсивной, число шагов отладки совпадает с числом итераций рекурсии, то есть пользовательские формулы проверяются на каждой итерации отдельно.

Выражения, записанные пользователем, задают требования к различным частям программы. Режим проверки формул позволяет определить, соответствует ли выполнение программы требованиям пользователя или вычислить интересные для пользователя выражения, не внося изменений в саму программу.

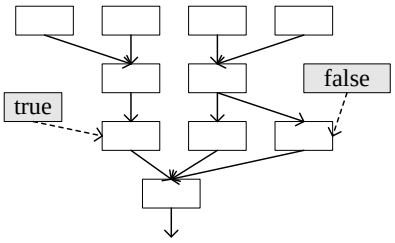


**Рис. 1**. Разметка вершин графа в режиме проверки формул

**3. Отладка над интервальными оценками входных аргументов**

Кроме пользовательских выражений в режиме проверки формул при отладке может применяться спецификация начальных данных функционально-потоковой параллельной программы, описывающая вид входных данных программы посредством стандартных данных, таких как списки, строки и числа, и специальных утверждений и констант. Множество утверждений спецификации выглядит следующим образом:

~unknownnumber – неизвестное число;

~unknownbool – неизвестное логическое значение (ложь или истина);

~gt A – число большее чем указанное число A;

~lt A – число меньшее чем указанное число A;

~ge A – число больше либо равное указанному число A;

~le A – число меньше либо равное указанному число A;

~A interval B – число лежащее в указанном интервале [A, B].

Спецификация начальных данных может иметь, например следующий вид: (5, ~lt 0, ~unknownbool) – список, первый элемент которого равен пяти, второй меньше нуля и третий является ложью или истиной, либо (~unknownnumber, 0) – список, первый элемент которого неизвестное число, а второй ноль.

Спецификация в виде интервальных оценок может задаваться пользователем, как для входных данных программы, так и для подстановки в спецификацию на графе.

При вычислениях над интервальными оценками не используются правила логического вывода или преобразование формул. Вместо этого применяется набор правил, созданный для каждого отдельного оператора функционально-потокового параллельного языка, позволяющий для известных входных данных оператора, среди которых есть хотя бы одно, заданное утверждением спецификации, поставить в соответствие результат работы оператора. Выходное значение оператора может быть известным данным, конкретным числом, строкой, списком или также являться утверждением спецификации, может являться комбинацией известных значений и утверждений спецификации. Набор правил не охватывает все возможные варианты, если во время отладки при рассмотрении оператора отсутствует правило для вывода его результата работы, результатом по умолчанию является константа спецификации ~unknown. При возникновении такой ситуации управление передается человеку. Пользователю предлагается самостоятельно ввести результат выполнения оператора над входными данными. Отказ от ввода означает, что результатом становится ~unknown, если же пользователь ввел результат, он безоговорочно принимается истинным и используется в дальнейших вычислениях.

Интервальные оценки могут применяться в пользовательских условиях. Условия записываются как выражения на самом языке программирования (без возможности вызова функций, описанных программистом), дополнительно в них можно использовать такие специальные константы:

- ARG – аргумент функции;
- NODE – значение той вершины-оператора графа функции, к которой добавлено пользовательское условие;
- NODE <натуральное число> - значение оператора с указанным номером, номера назначаются операторам автоматически перед началом отладки и отображаются в интерфейсе среды.

Таким образом пользовательское условие может быть к примеру таким ( (NODE,ARG):<, (NODE,~ 0 interval 1):!= ):*, то есть значение текущего оператора меньше чем аргумент функции и не совпадает с интервалом (0,1). При определении результата пользовательских выражений (условий) используется аналогичный подход, что и для вычисления самих операторов над интервальными оценками, то есть если утверждения спецификации отсутствуют в выражении, результат вычисляется интерпретатором, иначе происходит поиск в наборе правил, если поиск не дает результата происходит запрос к пользователю. Поэтому возможна и такая ситуация, когда результат введенного пользователем условия определяется им самим.

Представленная особенность отладки позволяет пользователю проследить выполнение программы над обобщенными входными данными, заданными посредством утверждений спецификации. Анализ процесса и результата выполнения программы над обобщенными начальными данными производится самим пользователем или отслеживается с помощью дополнительных пользовательских условий, приписанных к произвольным операторам программы.

**4. Пример отладки с подстановкой интервальных значений**

Функция Abs (рис. 2) получает число P и вычисляет его модуль.

Abs << funcdef P { ({P:-}, P): [(P,0):(<,>=) :?]:. >> return }

Пусть спецификацией начальных данных функции является ~lt 0, то есть P число меньшее нуля. Результат отладки функции над указанной интервальной оценкой входных значений будет следующим (рис. 2).

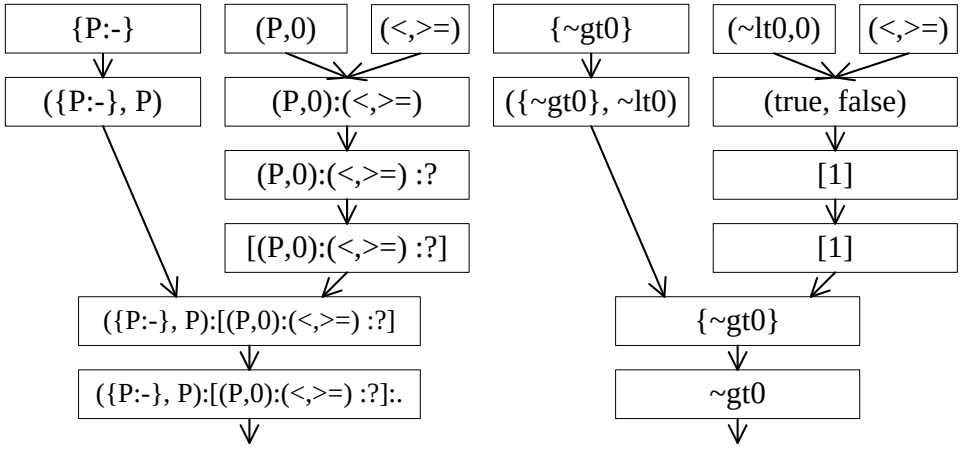


**Рис. 2.** Графы функции Abs с операторами и вычисленными значениями для аргумента ~lt0

Отладка в данном случае пройдет полностью автоматически, без запросов к пользователю и покажет, что функция Abs, получающая на входе отрицательное число, вычисляет число положительное.

Если запустить отладку функции Abs, указав спецификацию входного аргумента как ~ge 0, число большее либо равное нулю, аналогично будет показано, что результат выполнения функции станет ~ge 0. То есть отладка подтвердит, что функция вычислит число положительное и при положительном аргументе.

Не при всякой спецификации входных данных возможна полностью автоматическая обработка интервальных констант, так отладка с начальными данными вида ~gt 0 или ~le 0 сформирует запрос к пользователю на операторе «:?».

**5. Заключение**

Режим проверки формул позволяет задать и проверить как для конкретных начальных данных, так и для их интервальных оценок, пользовательские требования к процессу вычисления функционально-потоковой параллельной программы. Отладка над аргументами, заданными интервальными оценками, позволяет оперировать данными в обобщенном виде, получать интервальные оценки вычислений программы и при подключении пользовательских условий, прописывать спецификацию программы, как набор требований к произвольным узлам на ее графе. Такие возможности определяют развитый и эффективный инструментарий для выявления логических ошибок и исследования корректности программы.