Условные операторы: чуть теории для реализаторов

Н. Н. Непейвода email nnn@nnn.botik.ru

Freesoft 2024

Постановка проблемы

Гномы и эльфы

Огурец, помещённый в правильный рассол, хочешь не хочешь, станет солёным.

Бакланов пгниу

Ничего, через сто лет, может быть, поймёте.

Алгебраист, читающий теорию категорий информатикам.

Если вы надеетесь, что вас здесь вылечат — вы неправы. Мы сойдём с ума вместе.

Кащенко урфу

Примерно так

В своей семье, где знают и теорию, и практику, мы порою называем теоретиков эльфами, а практиков дворфами. И у тех, и у других есть свой тайный язык, а общаться они могут лишь на общем, из-за этого понимают друг друга плохо.

Пример

Напомним некоторые принципы понимания практиком теоретика (Вестник Удмуртского университета, 2007, №1, стр. 251-268).

1. Если теоретик говорит, что это в принципе возможно, понимайте: это практически невозможно, нужно уточнить задачу.
2. Если теоретик говорит, что это не имеет значения, понимайте: это имеет первостепенное значение для реализации.
3. Если вы считаете, что теоретики занимаются сферическими корнями в вакууме, говорить с ними бесполезно. Если вы осознали, что их мир другой, но имеет множество выходов на практику, надейтесь на успех.

Пример

Приведём два простейших примера перевода с эльфийского на гномий. Когда теоретик говорит, что операция коммутативна и ассоциативна (линейная логика Жирара), то практик должен насторожиться и понять, что у него есть прекрасные возможности распараллеливания, организации распределённого поиска и оптимизации. Ведь коммутативность и ассоциативность означает, что отдельные действия независимы, и их можно упорядочить или выполнять как угодно.

Если эльф говорит, что моделью является [неупорядоченное] множество, гном должен отложить кирку и задуматься сначала, как упорядочить или упорядочивать в процессе элементы своего представления.

Погружение в магму

Если же есть коммутативность без ассоциативности (эльф назовёт это коммутативной магмой), то возникает неупорядоченное дерево, в котором можно так же распределить или оптимизировать движение, или же наоборот, быстро, не задумываясь, написать главный цикл или рекурсию, и заняться более интересными вещами.

Некоторые соображения

Заметим, что всё время здесь появляется возможность распределённых вычислений, а свободные распределённые вычисления — это одно из направлений, где свободное программное обеспечение необходимо для слаженной работы волонтёров и решения трудных либо неразрешимых задач.

Некоторые соображения

Помимо выбора фрагментации исходного алгоритма, важной задачей является тактика работы. Наличие смотрящего-человека с несколькими простыми стратегиями действий в зависимости от доступного в данный момент множества волонтёров и полученных ранее результатов, позволяет с минимальными затратами труда и мозгов повысить эффективность в несколько раз. Так что тролли (или огры) тоже нужны в маленьких количествах.

Личный опыт

Есть ещё один важный принцип работы с теоретиками.
Если эльф говорит, что задача неразрешима или доказуемо трудна, то либо ограничьте аппетиты, выделив её часть, которая действительно нужна, либо поинтересуйтесь насчёт алгоритмов приближённого или вероятностного неточного решения.
Здесь важно вовремя поинтересоваться у эльфов, какие есть результаты по приближённым решениям. И чётко понять: что важнее, получить правильное решение в большем количестве случаев или избежать ошибочных решений. Если вы участвуете в олимпиаде по решению неразрешимых задач, очевиден второй выбор.

Верификация

В случае ограничения наилучшее разобраться с теоретиком, какие же подзадачи можно *в принципе* решить. Например, задача верификации программы доказуемо не проще, чем задача построения её с самого начала, что практически означает, что текст уже существующей программы будет чаще всего лишь мешать. Но задача верификации по типам данных или по областям изменения вполне решаемы и здесь текст программы помогает. Именно таковы задачи верификации протоколов.

И к постулатам СПО

Так что приходим к выводу Капитана Очевидность: проверить по настоящему можно лишь программу с открытым кодом, после чего ещё раз обращаемся к логикам и выясняем, что текста программы недостаточно в общем случае, нужно ещё иметь открытую спецификацию. И если это всё есть, гном может брать свою кирку и начинать долбахать с большой надеждой на успех либо на обнаружение ловушек. Заметим, что в случае общей задачи верификации открытая спецификация помогает слабо (зато уж точно не мешает!), но зато она помогает аккуратно пересоздать подозрительную программу с самого начала.

И к постулатам СПО

А приближённый алгоритм часто прекрасно ложится на распределённые вычисления. Заметим только, что, если вы реализовали недетерминированность либо неточность псевдослучайным алгоритмом, то на самом деле это детерминированный алгоритм, и вы никакого выигрыша не получите.

Инопланетяне

По моему опыту, ценнейшим источником может стать конструктивная математика, порою подсказывающая нетривиальные чисто практические решения (например, отреагировать на вопрос пользователя, что программа не сработала, советом чуть-чуть пошевелить в любую сторону данные, а не бросаться исправлять очередную ошибку).
Но тут возникает ещё более трудная задача перевода с инопланетного языка на гномий. Можно было бы много интересного здесь рассказать, но это потребовало бы часового доклада с интенсивной дискуссией. Так что займёмся более простым случаем взаимодействия теории и практики: условные операторы.

Условные операторы

Четырёхзначность

Адекватный анализ условных вычислений требует обработки ошибок [3]. Поэтому логика здесь четырёхзначная. T и F нормальные значения, минимально требуемые для условных вычислений. U ошибка, которую можно попытаться обработать и продолжить вычисления, $\perp$ крах.
Традиционный условный оператор
if A **then** M **else** N **fi**, как показал Маккарти, является базисом алгоритмических вычислений над произвольными абстрактными типами данных.

Маккартизм

Рекурсивные схемы, состоящие из нескольких определений вида

$$f_i(\vec{x}) \leftarrow \textbf{if } A(\vec{x}) \textbf{ then } M(\vec{f}, \vec{x}) \textbf{ else } N(\vec{f}, \vec{x}) \textbf{ fi}$$

дают одно из самых удобных, мощных и абстрактных понятий вычислимости. Они создавались для случая последовательных вычислений, где единственный способ преодолеть ошибку — защититься от её появления соответствующим условием. Поэтому ошибка в условии сразу переносится на ошибку во всём операторе.

Дейкстрин

Дейкстра заметил, что, если задана спецификация программы, то более адекватен условный оператора вида[1]

$$\textbf{if } A_1 \to S_1 \,\square\, \cdots \,\square\, A_n \to S_n \textbf{ fi},$$

выполняемый недетерминированно, поскольку в случае истинности нескольких условий A_i нам безразлично, какую из возможностей выбрать.

[1] В оригинальных обозначениях Дейкстра.

Пришёл реализатор и всё опошлил

Но при реализации данный оператор обычно детерминируют, что дополнительно затрудняет верификацию, проверку и особенно модификацию программ: почему это A_i стоит впереди A_j? Зато на распределённые вычисления дейкстровский оператор ложится прекрасно: задать волонтёрам свои $A_i \to S_i$ и организовать «соревнование»: кто первым найдёт и вычислит удачный вариант?

Хорошо забытое старое

Ещё одно очевидное и используемое, начиная с Фортрана, расширение условного оператора, когда управляющих значений не два $\{T, F\}$, а несколько. Такие переключатели легко обобщаются до дейкстровских и поддаются соревновательным вычислениям.

Логика Маккарти

Логика обычного оператора **if** также была описана Маккарти, и её называют логикой Лиспа. Она по умолчанию используется в большинстве реализаций логических связок, определяемых через **if**.

$$A \vee B \leftarrow \textbf{if } A \textbf{ then } T \textbf{ else } B \textbf{ fi} \quad (1)$$
$$A \wedge B \leftarrow \textbf{if } A \textbf{ then } B \textbf{ else } F \textbf{ fi}$$
$$\neg A \leftarrow \textbf{if } A \textbf{ then } F \textbf{ else } T \textbf{ fi}$$

Поэтому $a[i] > 0 \ \& \ i < 100$ потенциальная ошибка, а $i < 100 \ \& \ a[i] > 0$ правильное выражение.

Логика Маккарти 2

Выпишем таблицы истинности реализованных логических связок.

$A \wedge B$	$\mathfrak{t}$	$\mathfrak{f}$	$\mathfrak{u}$	$\bot$
$\mathfrak{t}$	$\mathfrak{t}$	$\mathfrak{f}$	$\mathfrak{u}$	$\bot$
$\mathfrak{f}$	$\mathfrak{f}$	$\mathfrak{f}$	$\mathfrak{f}$	$\mathfrak{f}$
$\mathfrak{u}$	$\mathfrak{u}$	$\mathfrak{u}$	$\mathfrak{u}$	$\mathfrak{u}$
$\bot$	$\bot$	$\bot$	$\bot$	$\bot$

$A \vee B$	$\mathfrak{t}$	$\mathfrak{f}$	$\mathfrak{u}$	$\bot$
$\mathfrak{t}$	$\mathfrak{t}$	$\mathfrak{t}$	$\mathfrak{t}$	$\mathfrak{t}$
$\mathfrak{f}$	$\mathfrak{t}$	$\mathfrak{f}$	$\mathfrak{u}$	$\bot$
$\mathfrak{u}$	$\mathfrak{u}$	$\mathfrak{u}$	$\mathfrak{u}$	$\mathfrak{u}$
$\bot$	$\bot$	$\bot$	$\bot$	$\bot$

(2)

Различение ошибок

Учитывая фатальность $\bot$, столь же естественна для реализации смешанная логика: Маккарти + слабый Клини, соответствующая определению $\vee$ и $\wedge$ без констант:

$$A \vee B \leftarrow \textbf{if } A \textbf{ then } A \textbf{ else } B \textbf{ fi} \tag{3}$$

$$A \wedge B \leftarrow \textbf{if } A \textbf{ then } B \textbf{ else } A \textbf{ fi}$$

$A \wedge B$	$\mathfrak{t}$	$\mathfrak{f}$	$\mathfrak{u}$	$\bot$
$\mathfrak{t}$	$\mathfrak{t}$	$\mathfrak{f}$	$\mathfrak{u}$	$\bot$
$\mathfrak{f}$	$\mathfrak{f}$	$\mathfrak{f}$	$\mathfrak{f}$	$\bot$
$\mathfrak{u}$	$\mathfrak{u}$	$\mathfrak{u}$	$\mathfrak{u}$	$\bot$
$\bot$	$\bot$	$\bot$	$\bot$	$\bot$

$A \vee B$	$\mathfrak{t}$	$\mathfrak{f}$	$\mathfrak{u}$	$\bot$
$\mathfrak{t}$	$\mathfrak{t}$	$\mathfrak{t}$	$\mathfrak{t}$	$\bot$
$\mathfrak{f}$	$\mathfrak{t}$	$\mathfrak{f}$	$\mathfrak{u}$	$\bot$
$\mathfrak{u}$	$\mathfrak{u}$	$\mathfrak{u}$	$\mathfrak{u}$	$\bot$
$\bot$	$\bot$	$\bot$	$\bot$	$\bot$

(4)

С точки зрения чистой логики и алгоритмики она ничего нового не даёт, но для реализации она полезна как дополнительная защита от эффектов краха.

Стандартные случаи

В дальнейших определениях три простейших случая, когда условие даёт определённый результат или аварию, совпадают, поэтому мы их выпишем однократно:

а) **if** $\bot$ **then** B **else** C **fi** $= \bot$.

б) **if** T **then** B **else** C **fi** $= B$;

в) **if** F **then** B **else** C **fi** $= C$.

В стандартном условном операторе добавляется ещё

г) **if** U **then** B **else** C **fi** $= U$.

Из Сибири с приветом

Группа Поттосина в ВЦ СО АН СССР заметила, что при параллельных вычислениях можно интерпретировать условный оператор по-другому. Вычисляем условие и обе альтернативы параллельно. Если условие никак не вычислится до конца либо даёт исправимую ошибку, но обе альтернативы дали один и тот же результат, то его и выдаём.

$$\textbf{if } A \textbf{ then } M \textbf{ else } M \textbf{ fi} = M$$

Новые логики

С логической точки зрения это решение красивое и даёт новую логику: сильный Клини + Маккарти

$A \wedge B$	$\mathfrak{t}$	$\mathfrak{f}$	$\mathfrak{u}$	$\bot$
$\mathfrak{t}$	$\mathfrak{t}$	$\mathfrak{f}$	$\mathfrak{u}$	$\bot$
$\mathfrak{f}$	$\mathfrak{f}$	$\mathfrak{f}$	$\mathfrak{f}$	$\bot$
$\mathfrak{u}$	$\mathfrak{u}$	$\mathfrak{f}$	$\mathfrak{u}$	$\bot$
$\bot$	$\bot$	$\bot$	$\bot$	$\bot$

$A \vee B$	$\mathfrak{t}$	$\mathfrak{f}$	$\mathfrak{u}$	$\bot$
$\mathfrak{t}$	$\mathfrak{t}$	$\mathfrak{t}$	$\mathfrak{t}$	$\bot$
$\mathfrak{f}$	$\mathfrak{t}$	$\mathfrak{f}$	$\mathfrak{u}$	$\bot$
$\mathfrak{u}$	$\mathfrak{t}$	$\mathfrak{u}$	$\mathfrak{u}$	$\bot$
$\bot$	$\bot$	$\bot$	$\bot$	$\bot$

(5)

И полезную новую связку надёжных вычислений с дублированием, обобщающуюся на нелогические значения:

$A \parallel B$	$\mathfrak{t}$	$\mathfrak{f}$	$\mathfrak{u}$	$\bot$
$\mathfrak{t}$	$\mathfrak{t}$	$\mathfrak{u}$	$\mathfrak{u}$	$\bot$
$\mathfrak{f}$	$\mathfrak{u}$	$\mathfrak{f}$	$\mathfrak{u}$	$\bot$
$\mathfrak{u}$	$\mathfrak{u}$	$\mathfrak{u}$	$\mathfrak{u}$	$\bot$
$\bot$	$\bot$	$\bot$	$\bot$	$\bot$

(6)

Голосование

Наличие $\|$ позволяет в одну строчку доказать, что рекурсивные схемы с новосибирской логикой сильнее обычных.
Через $\|$ очевидным образом определяется связка, соответствущая n-кратному дублированию счёта, например, трёхкратный счёт: $A \parallel (B \parallel C)$, и связки, соответствующих голосованию, например, 2 из 3:

$$V(A, B, C) \leftarrow (A \parallel B) \vee (A \parallel C) \vee (C \parallel B) \qquad (7)$$

Существенно, что дизъюнкция берётся в версии сильной клиниевской логики.

Реакция гнома

Но гном здесь должен задуматься и скорее всего отвергнуть. Во-первых, это решение уменьшает шансы обойти ошибку, так как вычисляются все три части условного оператора. Во-вторых, оно плохо масштабируется на дейкстровский **if** и переключатели.

Максимальный if

Если чисто логически попытаться развить новосибирскую идею, то приходим к максимальному $\mathbf{if}_3$. Если условие не вычисляется или даёт устранимую ошибку, и все альтернативы дают одно и то же значение либо устранимую ошибку, то принимаем выдаваемое значение.

a) $\mathbf{if}_3\ U\ \mathbf{then}\ T\ \mathbf{else}\ F\ \mathbf{fi}_3 = U$;
b) $\mathbf{if}_3\ U\ \mathbf{then}\ F\ \mathbf{else}\ T\ \mathbf{fi}_3 = U$;
c) $\mathbf{if}_3\ U\ \mathbf{then}\ T\ \mathbf{else}\ T\ \mathbf{fi}_3 = T$;
d) $\mathbf{if}_3\ U\ \mathbf{then}\ F\ \mathbf{else}\ F\ \mathbf{fi}_3 = F$;
e) $\mathbf{if}_3\ U\ \mathbf{then}\ B\ \mathbf{else}\ C\ \mathbf{fi}_3 = \bot$, если хоть одно из B, C есть $\bot$;
f) $\mathbf{if}_3\ U\ \mathbf{then}\ B\ \mathbf{else}\ U\ \mathbf{fi}_3 = B$;
g) $\mathbf{if}_3\ U\ \mathbf{then}\ U\ \mathbf{else}\ C\ \mathbf{fi}_3 = C$.

Максимальный if: оценка

Этот вариант использован при управлении вычислениями по переполнениям, но он так же плохо обобщается на случаи дейкстровского оператора, переключателей и распределённых вычислений. Зато с логической точки зрения он даёт логику Лукасевича (сильная Клини + своя импликация)

$A \supset_L B$	$\mathfrak{t}$	$\mathfrak{f}$	$\mathfrak{u}$	$\bot$
$\mathfrak{t}$	$\mathfrak{t}$	$\mathfrak{f}$	$\mathfrak{u}$	$\bot$
$\mathfrak{f}$	$\mathfrak{t}$	$\mathfrak{t}$	$\mathfrak{t}$	$\mathfrak{t}$
$\mathfrak{u}$	$\mathfrak{t}$	$\mathfrak{u}$	$\mathfrak{t}$	$\bot$
$\bot$	$\bot$	$\bot$	$\bot$	$\bot$

(8)

и функционально полон. Тем самым рекурсивные схемы с ним ещё сильнее.

Пролог

Ещё один вариант обработки ошибок — управление по неудачам, как в Прологе. Если условие дало ошибку, выбираем первую из альтернатив, давшую определённый результат.
Этот вариант $\mathbf{if}_4$ ориентирован на последовательное исполнение и также функционально полон с логической точки зрения. Он отлично обобщается на переключатели. Дейкстровский вариант такого оператора соответствует стандартной реализации **case** в языках, начиная с Лисп.

Логические эффекты

if$_4$ соответствует дисциплине, где $\mathfrak{u}$ понимается как отсутствие определённого решения, а не как ошибка. Первым вычисляется A, если оно дало решение, без дальнейших вариантов вычисляется соответствующая альтернатива. Если решения не нашлось, вычисляется B, если и оно не дало решения, то C. Он выразим через **if**$_3$:

$$\begin{aligned}\textbf{if}_4 \leftarrow \textbf{if}_3\ \neg j_U(A)\ \textbf{then if}_3\ A\ \textbf{then}\ B\ \textbf{else}\ C\ \ \textbf{fi}_3\\ \textbf{elif}_3\ \neg j_U(B)\ \textbf{then}\ B\ \textbf{else}\ C\ \textbf{fi}_3\end{aligned} \tag{9}$$

Здесь $j_U(A)$ одноместная логическая функция, принимающая значение $\mathfrak{t}$ на $\mathfrak{u}$, $\bot$ на $\bot$, и $\mathfrak{f}$ иначе.

Логика 4

Он даёт логику Розоноэра

$$\begin{array}{c|c|c|c|c} A \supset_R B & \mathfrak{t} & \mathfrak{f} & \mathfrak{u} & \bot \\ \hline \mathfrak{t} & \mathfrak{t} & \mathfrak{f} & \mathfrak{u} & \bot \\ \mathfrak{f} & \mathfrak{t} & \mathfrak{t} & \mathfrak{t} & \mathfrak{t} \\ \mathfrak{u} & \mathfrak{t} & \mathfrak{f} & \mathfrak{t} & \bot \\ \bot & \bot & \bot & \bot & \bot \end{array} \tag{10}$$

Другое отрицание

Возникает также интересная модификация импликации Розоноэра

$$\begin{array}{c|c|c|c|c} A \supset_{R*} B & \mathfrak{t} & \mathfrak{f} & \mathfrak{u} & \bot \\ \hline \mathfrak{t} & \mathfrak{t} & \mathfrak{f} & \mathfrak{u} & \bot \\ \mathfrak{f} & \mathfrak{t} & \mathfrak{t} & \mathfrak{t} & \mathfrak{t} \\ \mathfrak{u} & \mathfrak{t} & \mathfrak{f} & \mathfrak{t} & \mathfrak{t} \\ \bot & \bot & \bot & \bot & \bot \end{array} \tag{11}$$

единственная, которая может частично преодолеть эффект $\bot$.

Импликация Розоноэра оказывается с вычислительной точки зрения плохо совместима со стандартным отрицанием и порождает своё:

$$\begin{array}{c|c|c|c|c} A & \mathfrak{t} & \mathfrak{f} & \mathfrak{u} & \bot \\ \hline \neg_R A & \mathfrak{f} & \mathfrak{t} & \mathfrak{f} & \bot \end{array} \tag{12}$$

Вывод

Таким образом, теоретический анализ в данном случае приводит к двум выводам. Во-первых, подтверждается целесообразность реализации стандартного либо прологовского варианта обработки условий. Во-вторых, в случае стандартного варианта традиционная реализация **case** является одной из *подпорок*, вставляемых для согласования с условностями реализации и затрудняющих анализ и модификацию программы.

Анализ вывода

Но саркастическая ошибка теоретического анализа состоит в том, что научной истины нет. Есть лишь условный вывод, базирующийся на массе допущений, многие из которых эльфы и гномы не высказывают вслух, причём умолчания разные, и поэтому они особенно коварны. В частности, если перейти к рекурсивным схемам, то здесь наиболее мощный результат, а также наибольшие возможности оптимизации, проверки и модификации даёт логически полный третий **if**. Прологовский формальную мощь даёт ту же, но сложность определений становится больше, и, соответственно, возможности оптимизации и модификации хуже.

О реализуемости

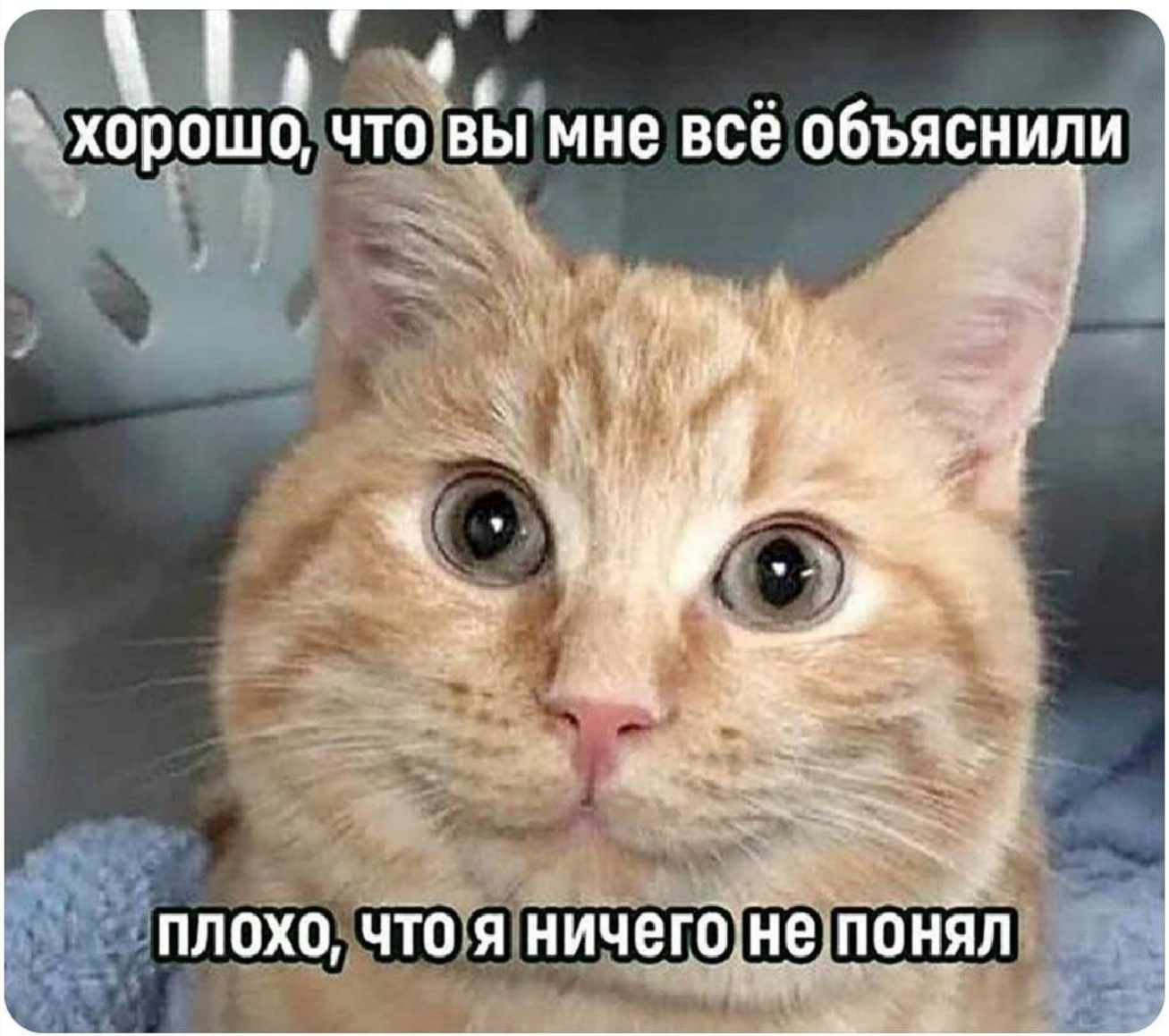
И, наконец, заметим, что непосредственно реализовать недетерминированность трудно, если вычисления не распределённые. Но гном может адекватно трактовать недетерминированность как разрешение детерминировать данное место любым способом, и использовать недетерминированный вариант как спецификацию либо даже разрешить его в языке с указанием, что транслятор имеет право детерминировать данный фрагмент любым способом, и оставить на усмотрение оптимизатора. В забытом ныне языке Алгол-68 [4] было введено понятие совместного исполнения, когда операторы разделялись запятыми, что означало позволение транслятору вычислять их последовательно в любом порядке либо даже параллельно.

[1] Непейвода Н. Н. Математик и прикладник: о взаимо(не)понимании
Вестник Удмуртского университета, 2007, №1, стр. 251-268.

[2] Emmanuel Beffara. Introduction to linear logic. Master. Italy. 2013. cel-01144229

[3] Непейвода Н. Н. Логики условных вычислений с ошибками. // Принята в журнал «Логические исследования» 2024.

[4] Пейган Ф. Дж. Практическое руководство по Алголу 68 = A Practical Guide to ALGOL 68 / пер. с англ. А.Ф. Рара, под ред. А. П. Ершова. — М.: Мир, 1979. — 240 с.



Вопросы
&
Открыт для дискуссии в кулуарах