СИСТЕМНАЯ ИНФОРМАТИКА

7

СИСТЕМНАЯ ИНФОРМАТИКА 7

ПРОБЛЕМЫ ТЕОРИИ И МЕТОДОЛОГИИ СОЗДАНИЯ ПАРАЛЛЕЛЬНЫХ И РАСПРЕДЕЛЕННЫХ СИСТЕМ

СБОРНИК НАУЧНЫХ ТРУДОВ

Под редакцией
доктора физико-математических наук, профессора **И. В. Поттосина**

НОВОСИБИРСК
"НАУКА"
2000

СИСТЕМЫ ПРОГРАММИРОВАНИЯ В ОГРАНИЧЕНИЯХ (ОБЗОР)

Д. М. Ушаков, В. В. Телерман

1. ВВЕДЕНИЕ

В научных исследованиях, проводимых в рамках дисциплины *искусственный интеллект*, важное место занимают проблемы разработки универсальных методов решения задач. Часто результатом таких исследований является создание новых языков программирования высокого уровня. Многие концепции и подходы, предложенные в этих языках, находят широкое применение в промышленном программировании. Примерами могут служить разработанные исследовательскими группами по искусственному интеллекту языки Prolog, Lisp и Smalltalk, на основе которых зародились парадигмы логического, функционального и объектно-ориентированного программирования соответственно. К настоящему времени произошло становление еще одной парадигмы программирования, которое носит название *программирование в ограничениях*.

Программирование в ограничениях базируется на результатах исследований методов решения *задач удовлетворения ограничений*. Задачи этого класса определяются совокупностью *ограничений*, связывающих между собой значения различных переменных. Решением задачи является такое означивание переменных, которое *удовлетворяет* всем заданным ограничениям. Большинство задач, возникающих в различных областях науки и техники, можно отнести к этому классу.

Существует достаточно большое количество языков программирования в ограничениях. К данному классу относятся языки программирования, обладающие средствами спецификации и решения задач удовлетворения ограничений. Как правило, эти языки являются расширениями существующих языков программирования. Такому подходу присущи следующие недостатки: средства спецификации задач удовлетворения ограничений определяются выразительными возможностями базового языка,

которые зачастую не подходят для этих целей, а семантика расширенного языка недостаточно прозрачна. Таким образом, становится актуальной проблема создания новых языков программирования в ограничениях, обладающих мощными средствами спецификации и решения задач удовлетворения ограничений и сохраняющих концептуальную целостность и прозрачность.

В нашей стране исследованием, разработкой и развитием средств программирования в ограничениях занимаются в основном сотрудники Российского научно-исследовательского института искусственного интеллекта (г. Москва) и его Новосибирского филиала. Они работают в тесном сотрудничестве с лабораторией искусственного интеллекта Института систем информатики им. А.П. Ершова СО РАН (г. Новосибирск). Этими двумя коллективами, помимо других программных систем, создан программный комплекс НеМо+ [1–3], включающий в себя развитые средства спецификации и решения задач удовлетворения ограничений.

Материал работы построен следующим образом. В разд. 2 дается определение задачи удовлетворения ограничений, показываются преимущества выделения таких задач в отдельный класс, приводятся примеры реальных задач этого класса. Исторический обзор работ в области программирования в ограничениях (разд. 3) знакомит со становлением данного научного направления и позволяет оценить успехи, достигнутые в этой области знаний. Классификация и анализ задач удовлетворения ограничений и алгоритмов для их решения, рассмотренные в разд. 4, показывают широту применимости методов программирования в ограничениях. В разд. 5 представлен обзор языков и систем программирования в ограничениях, включающий в себя краткий анализ большинства известных систем и их место в общей классификации. Указывается особое место комплекса НеМо+ в спектре этих систем. В заключении приводится сравнительная таблица всех рассмотренных систем программирования в ограничениях.

2. ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ И ПРОГРАММИРОВАНИЕ В ОГРАНИЧЕНИЯХ

Исследования в области искусственного интеллекта (ИИ) привели к созданию таких формальных средств представления и обработки знаний, как семантические сети, продукционные системы, фреймы, нечеткая математика, нейронные сети. Многочисленные промышленные интеллектуальные системы, созданные на базе этих средств, обосновывают необходимость проведения таких исследований, несмотря на скепсис, разделяемый большинством ИИ-исследователей, относительно возможности создания искусственного интеллекта как такового.

Несмотря на многочисленность разработанных ранее средств представления и обработки знаний, исследования в данном направлении продолжаются. В настоящее время фокус исследований в данной области переместился на разработку методов и средств удовлетворения ограничений и создания промышленных систем на их основе. Одно из объяснений этому состоит в том, что многие задачи, встречающиеся в разных областях, можно отнести к классу *задач удовлетворения ограничений.*

Определение 1. Задача удовлетворения ограничений, *или* ЗУО, — *это четверка* (D, R, V, C), *где*

- D — *множество* областей значений,
- R — *множество* отношений *различной арности над этими областями,*
- V — *множество* переменных, *с каждой из которых связана область ее возможных значений,*
- C — *множество* ограничений, *связывающих значения переменных из* V *посредством отношений из* R.

Решение *ЗУО* (D, R, V, C) — *это означивание переменных* V, *которое сопоставляет каждой переменной значение из ее области, такое, что для любого ограничения* $c \in C$ *выполняется соответствующее отношение над значениями связанных им переменных.*

Понятно, что под определение попадает достаточно широкий класс задач, которые различаются своими компонентами: областями значений переменных, видами отношений, способом задания ограничений. Может показаться, что введение нового типа задач не привносит ничего нового в теорию представления и обработки знаний, так как для решения конкретных видов таких задач можно использовать ранее полученные результаты в других областях. Тем не менее выделение задач удовлетворения ограничений в отдельный класс принесло следующие выгоды.

- Благодаря общему подходу к задачам такого типа были разработаны различные *кооперативные* методы их решения [4, 5], позволяющие применять к задачам комбинации известных методов. Еще важнее здесь появление ряда *универсальных* алгоритмов решения задач удовлетворения ограничений [6–9].
- Подход к спецификации задачи как задачи удовлетворения ограничений позволяет полностью избавиться от всех процедурных деталей спецификации. Вместо того чтобы специфицировать различные *алгоритмы*, позволяющие по входным данным вычислять результаты, нужно просто задать *модель* задачи. Оперировать такими *декларативными* спецификациями намного проще, чем спецификациями *императивными*, или алгоритмическими. Подход "модель вме-

сто алгоритма" позволяет поднять на качественно новый уровень средства представления и обработки знаний [10].

- Специфика задач удовлетворения ограничений позволила ввести новую компоненту знаний — *недоопределенность* [7, 11, 12]. Вместо того чтобы искать решение задачи, можно попробовать тем или иным способом *оценить* множество ее решений. Проекция такой оценки на значение каждой переменной дает оценку ее значения. Эта оценка трактуется как *недоопределенное* значение переменной, которое может доопределяться при поступлении новых данных (добавлении новых ограничений). Качественный скачок здесь достигается за счет того, что зачастую информация о допустимых значениях переменной является для инженера (ученого, экономиста) не менее (а иногда и более) важной, чем конкретное решение задачи. На основе такой информации он может принимать более грамотные и взвешенные решения, так как ему открываются новые взаимосвязи параметров задачи, которые обычно не являются очевидными даже для задач средней сложности.

Далее в разд. 4 мы разберем различные виды задач удовлетворения ограничений, а здесь приведем список известных проблем такого типа.

- *Моделирование.* Часто при проведении научных или инженерных расчетов требуется установить, как тот или иной параметр задачи влияет на другой параметр и наоборот, т. е. решить прямые и обратные задачи для одной и той же системы ограничений. Технология программирования в ограничениях привносит в эту область новые решения [13].
- *Проектирование.* Здесь ограничения представляют собой требования, предъявляемые к результирующему продукту. Этим продуктом может быть самолет, дом, электрический трансформатор [14] или музыкальная пьеса.
- *Планирование.* Планирование процесса работ во времени с учетом ресурсных затрат является типичной задачей удовлетворения ограничений (следование по времени, конфликт за общий ресурс). Программирование в ограничениях существенно повышает качество систем планирования [15].
- *Компьютерное зрение.* В качестве ограничений задаются различные образы, которые можно распознать в общей картине. В работе [16], посвященной машинному зрению, был впервые описан алгоритм *фильтрации*, являющийся прообразом алгоритма удовлетворения ограничений.
- *Человеко-машинные интерфейсы.* Здесь ограничения формулирует сам пользователь, а система подбирает удовлетворяющую им кон-

фигурацию графического диалога: размер и расположение окон, реакцию на курсор мыши и др. [17].

- *Мультиагентные системы.* Поведение интеллектуального агента удобно описывать в виде ограничений. Тогда каждый агент в процессе своего взаимодействия с другими агентами будет руководствоваться этими ограничениями. В работе [18] предлагается стройная концепция мультиагентных систем (называемая *технологией активных объектов*), которая опирается на аппарат программирования в ограничениях.

3. ОБЗОР РАБОТ В ОБЛАСТИ ПРОГРАММИРОВАНИЯ В ОГРАНИЧЕНИЯХ

Стало очевидным, что наиболее весомый вклад в становление и развитие программирования в ограничениях как самостоятельного научного направления внесли исследователи, занимающиеся проблемой представления знаний, логическим программированием и интервальными вычислениями. Однако необходимо признать, что у истоков данного направления стоят исследования в области машинной графики и обработки изображений. Один из первых интерактивных графических интерфейсов [19] решал геометрические ограничения. Отношения, заданные в виде уравнений или таблиц, были использованы в спецификациях задач для некоторых САПР, но они не привели к каким-либо теоретическим обобщениям. Самое первое обобщение, являющееся достаточно близким сегодняшнему понятию сети ограничений, было понятие *вычислительной модели*, предложенное Э.Х. Тыугу в 1970 г. [20]. Основная идея вычислительных моделей состоит в том, что задается модель, набор значений входных переменных и множество выходных переменных. Система автоматически синтезирует программу, вычисляющую значения выходных переменных. Замечательным свойством вычислительных моделей является то, что для одной и той же модели можно задавать различные наборы входных и выходных переменных, что приводит к синтезу различных программ. Например, на одной и той же модели треугольника можно решать следующие задачи:

- "зная *треугольник*, вычислить *площадь* по *сторонаА*, *сторонаВ*, *сторонаС* ";
- "зная *треугольник*, вычислить *уголАВС* по *площадь*, *сторонаА*, *сторонаС* ".

Один из основных методов определения совместности значений — алгоритм фильтрации [16] был разработан также в области обработки изображений.

Montanari [21], занимаясь исследованиями в той же области обработки изображений, первым применил алгебраический подход к сети ограничений. Он также первым стал рассматривать понятия совместности дуг и совместности путей сети ограничений.

Основополагающей работой в области удовлетворения ограничений, по-видимому, следует считать статью Mackworth [6], в которой впервые была предпринята попытка систематизировать накопленные к тому времени результаты. Алгоритмы (AC-1,2,3, PC-1,2), приведенные в этой работе, стали отправной точкой для исследователей в области удовлетворения ограничений. Следует отметить, что Mackworth рассматривал только конечные области и бинарные ограничения.

Идеи этих алгоритмов в различных видах впервые появились в работах Burstall [22], где исследуется метод решения задач зашифрованного сложения, и Fikes [23], в которой рассматривалось решение задач удовлетворения ограничений над булевыми (двухэлементными) областями. Мотивацией указанных работ послужила неэффективность решения задач удовлетворения ограничений над конечными областями значений традиционными методами перебора.

Mackworth рассматривал алгоритмы для решения ЗУО, в которых все области в D конечны, а каждое ограничение $c(x,y)$ связывает две переменные $x, y \in V$ с помощью бинарного отношения $r_c \subseteq D_x \times D_y$. Для простоты изложения предположим, что наряду с каждым ограничением $c(x,y)$ в C присутствует его зеркальный аналог $c'(y,x)$, где $r_{c'} = \{(b,a) \mid (a,b) \in r_c\}$. Алгоритмы предназначены для достижения *совместности по дугам сети ограничений*, т. е. сокращению области возможных значений D_x каждой переменной $x \in V$ таким образом, чтобы для любого ограничения $c(x,y)$ и значения $a \in D_x$ нашлось бы значение $b \in D_y$, совместное со значением a в смысле ограничения $c(x,y)$: $(a,b) \in r_c$. Рассмотрим вкратце принцип работы алгоритма AC-3. (Алгоритмы AC-1 и AC-2 являются менее эффективными вариациями на ту же тему.)

Алгоритм AC-3.

$Q \leftarrow C$;
пока $Q \neq \emptyset$
    выбрать и удалить любой $c(x,y)$ из Q;
    для каждого $a \in D_x$
        если не существует $b \in D_y$ такого, что $(a,b) \in r_c$, то
            удалить a из D_x;
    если изменилось D_x, то
        $Q \leftarrow Q \cup \{c(z,x) \mid z \neq x, z \neq y\}$;
конец.

Суть этого алгоритма состоит в следующем: если для некоторого значения a переменной x не существует ни одного совместного значения b

параметра y (в смысле ограничения $c(x,y)$), то оно удаляется из множества D_x допустимых значений переменной x. Сложность алгоритма AC-3 составляет $O(ed^3)$, где e — количество ограничений в C, а d — размер наибольшей области из D.

В 1986 г. в работе [8] было предложено развитие алгоритма AC-3, имеющее вычислительную сложность $O(ed^2)$, что, как нетрудно убедиться, является нижней оценкой сложности любого алгоритма достижения совместности по дугам. Этот алгоритм (названный AC-4) использует следующие две внутренние структуры данных:

1) для каждого значения $b \in D_y$ определяется множество пар (переменная, значение) ((x,a), где $x \in V$ и $a \in D_x$) таких, что значения a и b являются совместными в смысле ограничения $c(x,y)$:

$$S_{yb} = \{(x,a) | a \in D_x, c(x,y) \in C, (a,b) \in r_c\};$$

2) для каждой пары (x,a) хранится счетчик значений в $b \in D_y$, ее поддерживающих:

$$Counter((x,a),y) = \#\{(y,b) | (y,b) \in S_{xa}\}.$$

Работу алгоритма можно разделить на две части. Сначала для каждого ограничения $c(x,y)$ и для каждого значения $b \in D_y$ генерируется множество S_{yb} и инициализируются счетчики $Counter((x,a),y)$. Обнаруженные на этом этапе несовместные значения заносятся в очередь.

На втором этапе распространяются обнаруженные несовместности: несовместность пары (y,b) приводит к уменьшению на единицу значения счетчика $Counter((x,a),y)$ для всех $(x,a) \in S_{yb}$ таких, что $a \in D_x$. Пары (x,a), счетчики которых стали равными нулю, добавляются в очередь, а значение a удаляется из множества D_x.

Основной недостаток алгоритма AC-4 — его высокая пространственная сложность ($O(n^2d^2)$, где n — количество переменных в задаче, а d — размер наибольшей области).

В 1990-х годах были предложены другие алгоритмы достижения совместности: AC-5 [24], AC-6 [25], AC-7 [26], которые незначительно улучшали производительность AC-3 и AC-4.

Алгоритм AC-5 эффективно использует особенности некоторых классов ограничений (в частности, численных), но в общем случае он сводится к алгоритмам AC-3 или AC-4. Его временная сложность для специфических классов ограничений сводится к $O(n^2d)$.

Алгоритм AC-6 является развитием алгоритма AC-4. Его основное отличие от AC-4 состоит в следующем. Все области значений переменных предполагаются полностью упорядоченными. Очевидно, что для сохранения значения b в множестве D_y достаточно знать, что существует

хотя бы одно совместное с ним значение $a \in D_x$ в смысле ограничения $c(x, y)$. Поэтому вместо множества всех значений, поддерживающих данное, достаточно хранить только одно значение, например минимальное в смысле упорядочения D_y. В случае, если это минимальное значение становится несовместным и удаляется из множества D_y, ищется следующее минимальное совместное значение.

Множество S_{yb} определяется следующим образом:

$S_{yb} = \{(x, a) | c(x, y) \in C$ и b является минимальным значением в D_y таким, что $(a, b) \in r_c\}$.

Так как в алгоритме AC-6 в множестве S_{yb} хранится только одно значение, его пространственная сложность — $O(n^2 d)$, хотя временная сложность такая же, как и у AC-4 — $O(n^2 d^2)$.

Алгоритм AC-7 является дальнейшим развитием AC-6. Его специфика состоит в том, что он учитывает такие свойства ограничений, как иррефлексивность и коммутативность. AC-7 имеет следующие особенности:

1) не проверять совместность пары значений (a, b), если известен $b' \in D_y$ такой, что пара значений (a, b') совместна;
2) не проверять совместность пары значений (a, b), если известен $b' \in D_y$ такой, что пара значений (b', a) совместна;
3) не проверять пару значений (a, b), если:
   a) она была уже проверена;
   b) пара (b, a) была уже проверена;
4) алгоритм имеет пространственную сложность $O(ed)$.

У алгоритма AC-3 отсутствуют свойства 1, 2, 3a и 3b. У AC-4 — 1, 2, 3b и 4. У AC-6 отсутствуют свойства, вытекающие из коммутативности ограничений, 2 и 3b.

Наиболее универсальный подход к спецификации и решению ЗУО был предложен А.С. Нариньяни в 1980 г. Этот метод основывается на вычислительных моделях Тыугу и интервальных вычислениях и был назван методом *недоопределенных моделей* [11, 7]. В недоопределенных моделях наряду с дискретными множествами значений можно использовать интервальные представления вещественных чисел. В недоопределенных моделях был также предложен наиболее универсальный вычислительный алгоритм, который позволяет применять данный подход к самым различным классам задач.

Рассмотрим подробнее этот подход. В работах А.С. Нариньяни [12] и его учеников [27] введено понятие *обобщенной вычислительной модели* (*ОВМ*), являющееся развитием вычислительной модели Тыугу. В этой модели вместо переменной x, принимающей значения в области D_x, рассматривается недоопределенная переменная (н-переменная) $^*x$

с областью значений ${}^*D_x$, где ${}^*D_x$ — множество всех подмножеств D_x за исключением пустого. Следуя Тыугу [20], А.С. Нариньяни сопоставляет каждому ограничению $c(x, y, \ldots, z)$ его *интерпретацию* множеством функций, позволяющих по известным значениям переменных $x, y, \ldots, z$ вычислить значения остальных. Например, ограничение

$$e^x - 2y^2 + z + 7 = 0,$$

связывающее переменные x, y и z, интерпретируется тремя функциями:

$$\begin{aligned} x &:= \ln(2y^2 - z - 7), \\ y &:= \sqrt{\frac{e^x + z + 7}{2}}, \\ z &:= 2y^2 - e^x - 7. \end{aligned}$$

Однако в отличие от Тыугу, А.С. Нариньяни рассматривает эти функции над н-переменными, а именно, если $f^1, \ldots, f^n$ суть функции интерпретации ограничения $c(x_1, \ldots, x_n)$, то в обобщенной вычислительной модели им соответствуют функции

$${}^*f^i({}^*x_1, \ldots, {}^*x_n) = \{f^i(x_1, \ldots, x_n) \mid x_k \in {}^*x_k \text{ для всех } k \neq i\} \cap {}^*x_i.$$

Алгоритм вычислений в ОВМ рассматривается как параллельный асинхронный недетерминированный пошаговый процесс. Каждая функция интерпретации и каждая н-переменная в ОВМ находятся в одном из двух состояний: активном или неактивном (пассивном). Активная функция интерпретации становится пассивной после своего исполнения; н-переменная становится активной, если меняется ее недоопределенное значение.

На первом шаге все функции интерпретации всех ограничений считаются активными и исполняются. После исполнения формируется множество активных н-переменных и активируются все функции интерпретации, имеющие среди своих аргументов хотя бы одну активную н-переменную. На каждом следующем шаге вычисляются все функции интерпретации, которые стали активными на предыдущем шаге, и вновь строится множество активных н-переменных. Процесс заканчивается тогда, когда множество активных н-переменных становится пустым.

Для представления недоопределенных чисел А.С. Нариньяни предложил использовать интервалы значений, а для реализации функций интерпретации над ними — аппарат *интервальной математики* [28]. Им были предложены и другие *недоопределенные типы данных*, в частности недоопределенные множества и отрезки.

Представление множеств значений интервалами, на наш взгляд, явилось значительным прорывом в области алгоритмов удовлетворения ограничений, который, однако, остался незамеченным западными учеными. Одной из причин стало то, что А.С. Нариньяни использовал терминологию представления знаний и потоковых вычислений, которая в корне отличалась от терминологии логического программирования (которой пользовался Mackworth и др.). Другое объяснение того, что на протяжении почти десяти лет отечественные и зарубежные исследователи в области удовлетворения ограничений разговаривали на "разных"языках заключается в том, что предложенный механизм недоопределенных вычислений своей универсальностью существенно превосходил существующие алгоритмы удовлетворения ограничений. В то время как западные исследователи решали ЗУО на конечных областях значений, основные усилия разработчиков аппарата недоопределенных моделей были направлены на решение численных задач.

Использование вещественных интервалов в алгоритмах удовлетворения ограничений было "переоткрыто" несколько раз в работах Davis [29], Cleary [30] и Hyvönen [31], которые имели многочисленных последователей (Older и Vellino [32], Lhomme [33], Benhamou, McAllester и Van Hentenryck [34]). Большинство алгоритмов удовлетворения интервальных ограничений можно рассматривать как разновидности алгоритма AC-3. Алгоритм AC-3 в "чистом виде" служит для вычисления *совместных по дугам* множеств значений переменных ЗУО. Обобщения этого алгоритма, работающие с интервалами значений, вычисляют *оболочно-совместные* (hull-consistent) множества значений.

Рассмотрим это понятие подробнее. Следуя Mackworth, назовем ограничение $c(x, y, \ldots, z) \in C$ *совместным по дугам*, если для любой переменной $v \in \{x, y, \ldots, z\}$ и любого значения $a_v \in D_v$ существуют значения $a_x \in D_x$, $a_y \in D_y$. ..., $a_z \in D_z$ такие, что $(a_x, a_y, \ldots, a_v, \ldots, a_z) \in r_c$.

Предположим теперь, что области из D являются подмножествами вещественной прямой. *Вещественным интервалом* $[\underline{a}, \overline{a}]$ назовем множество $\{a \in \mathcal{R} \mid \underline{a} \leq a \leq \overline{a}\}$. Введем операцию аппроксимации *hull*, сопоставляющую каждому подмножеству вещественных чисел наименьший содержащий его интервал. Назовем ограничение $c(x, y, \ldots, z) \in C$ *оболочно-совместным*, если для любой переменной $v \in \{x, y, \ldots, z\}$

$$D_v = hull(D_v \cap \{a_v \in \mathcal{R} \mid (\forall u \neq v)(\exists a_u \in D_u)(a_x, a_y, \ldots, a_z) \in r_c\}).$$

Алгоритм достижения оболочной совместности сети ограничений является обобщением алгоритма AC-3. В нем вместо перебора пар несовместных значений используется *оператор сужения*, который по заданному ограничению $c(x, y, \ldots, z)$ сужает области $D_x, D_y, \ldots, D_z$ до оболочно-совместных областей.

В работе [34] было предложено другое понятие локальной совместности: *блочная совместность* (box-consistency), которое определялось в терминах интервальных расширений функций, задающих ограничение. Ограничение $c(x, y, \ldots, z) \in C$, имеющее вид $f(x, y, \ldots, z) = 0$ (где $f : \mathcal{R}^n \to \mathcal{R}$), называется *блочно-совместным*, если для любой переменной $v \in \{x, y, \ldots, z\}$

$$D_v = hull(D_v \cap \{a_v \in \mathcal{R} \mid \tilde{f}(D_x, D_y, \ldots, [a_v, a_v], \ldots, D_z) = [0, 0]\}),$$

где $\tilde{f}$ — естественное интервальное расширение функции f [28]. В [34] показано, что для достижения блочной совместности может быть использован интервальный метод Ньютона. Различные понятия интервальной совместности исследовались и в работе [33].

Вообще, задача нахождения совместных областей значений переменных, представимых в виде интервалов, широко исследуется в смежной дисциплине — интервальном анализе. Например, известен факт [35], что множество всех решений системы уравнений

$$\begin{cases} f_1(x_1, \ldots, x_n) = 0, \\ \ldots, \\ f_m(x_1, \ldots, x_n) = 0 \end{cases}$$

(где f_i — непрерывные рациональные функции) содержит в себе алгебраическое интервальное решение $\mathbf{x}_1, \ldots \mathbf{x}_n$ системы уравнений

$$\begin{cases} \tilde{f}_1(\mathbf{x}_1, \ldots, \mathbf{x}_n) = [0, 0], \\ \ldots, \\ \tilde{f}_m(\mathbf{x}_1, \ldots, \mathbf{x}_n) = [0, 0], \end{cases}$$

где $\tilde{f}_i$ — естественное интервальное расширение f_i.

Следует заметить, что для ограничений над вещественными числами были разработаны различные специализированные (неинтервальные) методы, изложенные в работах Тыугу [4] и Jaffar и др. [36].

Важный вклад в создание эффективных алгоритмов решения задач удовлетворения ограничений над конечными областями внесли работы [37-39]. В частности, Freuder впервые ввел понятие *распространения ограничений*. Его идея состоит в том, чтобы автоматически синтезировать новые ограничения возрастающей арности (начиная с арности 1). Максимальная арность синтезированных ограничений равна количеству переменных в сети. Новые ограничения добавляются в исходную сеть и участвуют в вычислениях. Замечательным свойством алгоритма распространения ограничений является то, что при удовлетворении ограничения максимальной арности мы получаем явное решение задачи. В [37] обоснована корректность данного алгоритма.

Еще одной важной областью, к которой были применены методы удовлетворения ограничений, стала область множеств. Удовлетворение ограничений над множествами рассматривалось в работах [11, 40–44].

Указанные работы были обобщены рассмотрением ограничений над разнородными областями [45–48] и имеют много общего с исследованиями, посвященными *абстрактной интерпретации*, в частности с классической работой [49]. В [45] семантика удовлетворения ограничений рассматривается именно с позиции абстрактной интерпретации. В одной из последних работ [50] показано, что несмотря на все многообразие алгоритмов удовлетворения ограничений, всех их можно рассматривать как экземпляры одного универсального алгоритма. Именно этот алгоритм и был предложен в пионерской работе А. С. Нариньяни [7] по недоопределенным моделям, правда, без обоснования его корректности. Таким образом, можно сказать, что за 20 лет исследования в области алгоритмов удовлетворения ограничений совершили полный виток по гегелевской спирали развития науки.

В работе Д. Ушакова [51] была закрыта брешь между недоопределенными моделями и алгоритмами достижения локальной совместности сети ограничений. В ней введено понятие недоопределенного расширения $^*D_v$ области значений D_v переменной v:

- $\{\emptyset, D_v\} \subseteq {}^*D_v \subseteq 2^{D_v}$,
- ${}^*a \cap {}^*b \in {}^*D_v$ для любых ${}^*a, {}^*b \in {}^*D_v$

и операция аппроксимации любого подмножества значений в недоопределенном расширении

$$app_v(X) = \bigcap_{X \subseteq {}^*a \in {}^*D_v} {}^*a.$$

Нетрудно видеть, что множество всех подмножеств конечной области значений, так же как и множество всех вещественных интервалов, является недоопределенным расширением. Недоопределенные значения ${}^*a_x \in {}^*D_x$, ${}^*a_y \in {}^*D_y$, ..., ${}^*a_z \in {}^*D_z$ называются *н-совместными* с ограничением $c(x, y, \ldots, z)$, если они являются аппроксимациями некоторых множеств значений, совместных с ограничением $c(x, y, \ldots, z)$. Интерпретация ограничения состоит в нахождении наибольших н-совместных с этим ограничением недоопределенных значений, которые включены в исходные недоопределенные значения. Алгоритм вычисления недоопределенных значений, н-совместных со всеми ограничениями ЗУО, аналогичен алгоритму, предложенному А. С. Нариньяни. Однако такое определение недоопределенного расширения и интерпретации ограничений позволяет доказать, что результат алгоритма никак не зависит от порядка интерпретации ограничений. Таким образом, алгоритм AC-3, равно как

и его интервальные модификации, является разновидностью алгоритма вычислений в недоопределенных моделях, предложенного в работе [7]. Следует отметить по крайней мере три западные работы, опубликованные в середине 1990-х годов, в которых были предложены конструкции, аналогичные недоопределенным расширениям: Caseau и Puget [45], van Emden [46], Benhamou [52].

Несмотря на достаточно молодой возраст программирования в ограничениях, уже появилось несколько неплохих обзоров исследований в данной области [9, 53–55]. Однако в первых трех обзорах полностью отсутствует упоминание о каких-либо российских работах. Статья Т. М. Яхно [55] является первым русскоязычным обзором программирования в ограничениях, в котором впервые предложена используемая в настоящее время русская терминология и достаточно подробно дано введение в программирование в ограничениях. В нашем обзоре мы предполагаем, что читатель знаком с основными понятиями из области программирования в ограничениях и постарались сделать его наиболее полным, т.е. учли работы как зарубежных, так и российских авторов.

4. КЛАССИФИКАЦИЯ И АНАЛИЗ РАЗЛИЧНЫХ АЛГОРИТМОВ УДОВЛЕТВОРЕНИЯ ОГРАНИЧЕНИЙ

Определимся понимать под алгоритмом удовлетворения ограничений любой алгоритм, который находит решение задачи удовлетворения ограничений или, по крайней мере, сужает область возможных значений переменных задачи. Конечно, тот или иной алгоритм применим только к небольшому подклассу задач. Начнем классификацию алгоритмов с классификации решаемых ими задач. Каждая задача удовлетворения ограничений имеет четыре составляющих (определение 1), мы разберем каждую из них отдельно.

4.1. Области значений переменных

Конечные области. Задачи с конечными областями значений переменных — традиционный объект исследований в ИИ. Для решения таких задач было предложено (и продолжает предлагаться) множество методов (Нильсон [56]): от перебора с возвратами до генетических алгоритмов.

Числовые области. Задачи с такими областями — основной объект исследований в математике, где разработано огромное количество алгоритмов для их решения. Отличительной особенностью алгоритмов

является их специализированность: в математике практически нет *универсальных* методов решения, которые были бы применимы к широкому классу задач.

Составные области. Речь идет о таких популярных у программистов-теоретиков областях, как множества, деревья (термы) или графы. Багаж накопленных здесь алгоритмов может успешно использоваться для решения задач удовлетворения ограничений над этими областями.

Иерархии классов. Иерархическая организация гетерогенных (разнородных) областей может служить областью для задач удовлетворения ограничений. Универсальные алгоритмы для работы с такими областями, как и кооперативные решатели этих задач с помощью разных методов, сейчас начинают занимать центральное место в области программирования в ограничениях [45, 48].

4.2. Виды отношений

Конечно, тот или иной вид отношения определяется в первую очередь природой областей значений. Например, над числовыми областями существуют арифметические отношения типа "сумма двух чисел равна третьему", над графами — отношение изоморфности, над множествами — включения и т. д. Однако можно выделить одно универсальное свойство отношения, а именно разделить отношения на *экстенсиональные* и *интенсиональные*.

Экстенсиональные отношения. Речь идет об отношениях над конечными областями, которые можно задать простым перечислением всех кортежей элементов областей, удовлетворяющих данному отношению. Конечно, так можно задать все отношения над конечными областями. Другим, более важным примером таких отношений могут служить таблицы в реляционных базах данных. Многие свойства моделируемого предмета или явления часто задаются именно таким образом. Эти отношения играют важную роль в области САПР (систем автоматизированного проектирования) и в области ГИС (геоинформационных систем). Алгоритмы удовлетворения ограничений, поддерживающие экстенсиональные отношения, находят здесь самое широкое применение.

Интенсиональные отношения. Имеются в виду отношения, заданные параметрически. Таковыми являются почти все отношения над числовыми областями. Например, отношение "сумма двух чисел равна третьему" является интенсиональным. Соответственно для их обработки требуется применять специальные методы удовлетворения ограничений.

4.3. Способ представления значений переменных

Традиционный программистский способ, связывающий с каждой переменной одно значение из ее области определения, не всегда применим при решении задач удовлетворения ограничений. Рассмотрим, каким образом представляют значения переменных различные алгоритмы удовлетворения ограничений.

Единичные значения. Первые методы удовлетворения ограничений работали именно с такими значениями переменных. В частности, в работе [4] описывается система, позволяющая решать задачи, имея только их декларативную спецификацию. Решение происходит методом распространения известных значений переменных к неизвестным. При этом наряду с простым распространением констант в выражениях используются специализированные методы для решения различных подклассов задач: системы линейных уравнений и др. Другой плодотворный метод, основанный на единичных значениях, описан в работе [36]. Суть метода состоит в поиске точных решений линейных подсистем задачи, после которого вычисленные значения подставляются в нелинейные части. Благодаря этому степень нелинейности снижается, что позволяет еще раз применить тот же самый метод. Кроме того, единичные значения переменных используются в методах *иерархического* удовлетворения ограничений, предложенных Borning и др. [57, 17].

Множественные значения: экстенсиональный способ. Как отмечалось ранее, качественный скачок в решении задач удовлетворения ограничений был достигнут при переходе от единичных к множественным значениям для каждой переменной. Первые работы на эту тему [6, 21] дали общее название таким алгоритмам: алгоритмы достижения локальной совместности.

Множественные значения: интенсиональный способ. Представляя множество значений простым перечислением его элементов, мы ограничиваем себя конечными областями. Настоящий прорыв в области программирования в ограничениях был совершен в работах [7, 29–31], где предлагался способ представления множества значений переменных в виде интервалов, для задания которых достаточно указать два элемента — нижнюю и верхнюю границы. Использование интервалов позволило расширить сферу применения классического алгоритма AC-3, предложенного Mackworth [6], на числовые области. Тем самым был получен новый универсальный алгоритм оценки множества всех решений произвольной системы алгебраических уравнений, который имеет самостоятельную ценность в подобласти вычислительной математики — интервальном анализе [28, 58, 59]. Развитием идеи интервалов значений

является мультиинтервальное представление вещественных чисел, предложенное в интервальном анализе А. Г. Яковлевым [60], а в программировании в ограничениях впервые реализованное В. В. Телерманом [61].

4.4. Язык спецификации ограничений

Способ задания ограничений отличается в различных алгоритмах удовлетворения ограничений. Рассмотрим различные возможности задания ограничений.

Простые атомарные формулы. Наиболее простой способ задания ограничения заключается в связывании переменных с помощью отношения. При этом для каждого ограничения указывается набор связываемых им переменных и отношение, которое должно выполняться над их значениями.

Атомарные формулы, составленные из термов. Часто предметную область удобно описывать в терминах *функций*. При этом результат одной функции может использоваться в качестве аргумента другой. Математики предложили специальный язык для задания таких конструкций — язык *термов*. Атомарные формулы, построенные над термами, часто используются для задания ограничений.

Произвольные формулы первого порядка. Логика предикатов первого порядка претендует на роль универсального языка представления знаний в ИИ. Поэтому усилия многих разработчиков систем представления и обработки знаний были направлены на создание универсальных программ для обработки предложений логики первого порядка. Конечно, в рамках решения задач удовлетворения ограничений исследователи стараются по мере возможностей обрабатывать ограничения, выраженные на языке первого порядка. Определенные успехи, достигнутые в этой области [35, 62], свидетельствуют о перспективах применения методов удовлетворения ограничений для решения таких задач.

4.5. Алгоритмы удовлетворения ограничений

Как показано в предыдущих разделах, алгоритмы прежде всего различаются степенью своей универсальности, или подклассами задач удовлетворения ограничений, к которым они применимы. Кроме того, не все алгоритмы находят *решение* задачи. Часть алгоритмов находит только некоторую *область* в пространстве возможных решений задачи, внутри которой лежит решение. Во многих алгоритмах размером такой области можно управлять за счет изменения времени счета. Наконец, некоторые

алгоритмы, вычисляя такую область, не гарантируют *существование* решения внутри этой области. Условимся называть *точными* алгоритмы, которые находят решение задачи. Алгоритмы, которые указывают область возможных решений задачи, назовем *приближенными*. При этом те из них, которые гарантируют существование решения внутри вычисленной области, будем называть *полными*. Напомним вкратце о наиболее популярных алгоритмах, используемых в различных системах программирования в ограничениях.

4.5.1. Точные алгоритмы

Следует сразу отметить, что специфика представления вещественных чисел на компьютере отметает возможность реализации точных алгоритмов для решения большинства задач удовлетворения ограничений над вещественными областями. Поэтому точные алгоритмы применимы только к конечным областям, а также к таким числовым областям, как целые и рациональные числа.

Метод проб и ошибок. Таким способом можно решать любую задачу удовлетворения ограничений. Идея состоит в означивании переменных и проверке выполнения всех ограничений. Если не все ограничения выполняются, нужно попробовать другое означивание переменных. Рано или поздно, для конечных задач решение будет найдено. Недостатком способа является его очевидная неэффективность: время решения задачи экспоненциально зависит от количества переменных.

Перебор с возвратами. Перебор с возвратами (бэктрекинг) является улучшением метода проб и ошибок. В этом способе мы по очереди означиваем каждую переменную и проверяем выполнимость ограничений на частичном означивании. В случае неудачи пробуем другое значение для последней означенной переменной. Если все возможные значения уже исчерпаны, следует вернуться к предыдущей переменной и попробовать означить ее и т. д. Перебор с возвратами достаточно хорошо работает на многих задачах удовлетворения ограничений с конечными областями, однако его трудоемкость по-прежнему экспоненциально зависит от числа переменных задачи. Конечно, перебор с возвратами, так же как метод проб и ошибок, работает только на задачах с конечными областями.

Другие переборные методы. За последние двадцать лет был предложен ряд методов перебора, повышающих эффективность бэктрекинга. К ним относится бэкмаркинг [63], бэкджампинг [64], проверка вперед [65], динамический бэктрекинг [66] и др. Эти методы становятся особенно эффективными в комбинации с методами достижения локальной совместности [67].

Метод Гаусса для систем линейных уравнений. Для решения задач с числовыми областями традиционно используются кардинально различные методы. В случае ограничений, заданных линейными уравнениями, эффективно работает метод Гаусса для решения систем линейных уравнений, имеющий полиномиальную сложность.

Симплекс-метод. Предложен для решения задач линейного программирования [68]. Несмотря на теоретическую экспоненциальную сложность, он доказал свою эффективность при решении конкретных задач. Этот градиентный метод часто используют и для нахождения произвольного (не обязательно оптимального) решения системы линейных неравенств.

4.5.2. Приближенные алгоритмы

Итерационные методы решения систем уравнений и их интервальные аналоги. В вычислительной математике разработан ряд методов для эффективного решения систем (линейных) уравнений. Данные алгоритмы находят приближенное решение, для которого можно определить критерий погрешности (как далеко лежит настоящее решение от найденного). Неудовлетворенность таким способом оценки решения, равно как и учет реалий конкретных задач, привел и к зарождению в рамках вычислительной математики новой науки — *интервального анализа.* Причиной создания интервального анализа послужило простое прагматическое наблюдение: в реальном мире мы не можем оценивать параметры задачи *точно*, мы можем только указать приблизительное распределение значения параметра. Наиболее простым способом задания такого распределения является указание *интервала* возможных значений. В интервальном анализе разработаны специальные методы для обработки таких интервальных значений [28, 59]. Многие известные методы решения систем уравнений (полношаговый итерационный процесс решения систем линейных уравнений, метод Ньютона для решения алгебраических уравнений и др.) были адаптированы к случаю интервальных параметров [35, 59].

Методы достижения локальной совместности. К ним относятся упомянутые в предыдущем разделе алгоритмы AC-1–AC-7, их интервальные аналоги, а также метод недоопределенных вычислений Нариньяни. Все они восходят к алгоритму, предложенному Waltz [16] для распознавания образов. Большинство алгоритмов достижения локальной совместности — это приближенные неполные алгоритмы с полиномиальной сложностью решения.

Методы кооперативного решения задач. Другим способом решения ЗУО является кооперативное решение задач: применение к одной и той же задаче (или к разным частям одной и той же задачи) нескольких специализированных методов. Теоретические и технологические аспекты кооперативных решателей задач исследовались в работах [4, 47, 69].

5. ЯЗЫКИ И СИСТЕМЫ ПРОГРАММИРОВАНИЯ В ОГРАНИЧЕНИЯХ

Как отмечалось в разд. 2, постановка задачи в виде системы ограничений привносит свою специфику в аппарат знаний. По сути, выделение такого класса задач и создание программных систем их спецификации и решения привело к появлению парадигмы программирования в ограничениях. Языки программирования, реализованные в рамках этой парадигмы, различаются прежде всего средствами задания ограничений, или, вообще говоря, задач удовлетворения ограничений. Простейший способ такого задания — *статическое* (декларативное) определение задачи. В то же время в различных приложениях требуется динамически решать различные задачи удовлетворения ограничений, и поэтому многие системы предлагают *динамические* средства спецификации. Речь идет о возможности переходить в процессе решения от одной формулировки задачи к другой, добавляя или удаляя ограничения.

Важной особенностью динамических средств спецификации является возможность запоминания состояния задачи с последующим откатом к нему в случае необходимости. Для представления динамики используется одна из наработанных парадигм программирования: императивное программирование (последовательное выполнение шагов с возможностью условного перехода, отметим также параллельное программирование), логическое программирование (унификация и откат), объектно-ориентированное программирование (реакция объектов на поступающие сообщения). Объектно-ориентированное программирование можно рассматривать как разновидность императивного. Здесь многие исследователи пошли по пути расширения какого-нибудь существующего языка той или иной парадигмы. Однако на этом пути не всегда ожидает успех, так как изменение сложившегося языка со стройной семантикой, как правило, сказывается на нем отрицательно. Кроме того, средства спецификации ограничений в таком языке ограничиваются возможностями базового языка, которых зачастую недостаточно для этих целей. Наиболее перспективные исследования в области программирования в ограничениях связаны с разработкой новых языков представления и обработки знаний. Отметим также, что не все исследователи пошли по этому пу-

ти до конца — многие остановились на создании специализированных библиотек для одного из существующих языков программирования.

Подводя итог изложенному, можно предложить следующую классификацию систем программирования в ограничениях. По способу спецификации ЗУО выделим следующие разновидности программирования в ограничениях:

- *"Чистое" программирование в ограничениях* использует статическую спецификацию задачи удовлетворения ограничений. Уровень такой спецификации может варьироваться от системы к системе.

- *Логическое программирование в ограничениях* использует для спецификации ЗУО средства языка логического программирования Пролог и его аналогов. Задача при этом может изменяться путем добавления в нее новых переменных и ограничений. При откате происходит возврат к предыдущему состоянию задачи.

- *Императивное (в том числе параллельное) программирование в ограничениях* использует обычные императивные конструкции для перехода между *состояниями*. Каждое состояние такой императивной программы характеризуется не только значениями переменных и потоком управления, но и *памятью ограничений* (constraint store). Память ограничений можно рассматривать как отдельный объект с двумя выделенными операциями: добавить новое ограничение в память и проверить выполнимость заданного ограничения на текущей памяти. Таким образом, память ограничений служит здесь чем-то вроде машины логического вывода, с помощью которой значительно упрощается программирование сложных интеллектуальных систем. Параллельные системы реального времени имеют ряд отличительных особенностей, которые мы разберем ниже. Отметим также, что к этому ряду примыкают продукционные системы, использующие технологию программирования в ограничениях [70].

По способу реализации различаются следующие группы систем:

- библиотеки стандартного языка программирования,
- расширения существующего языка программирования,
- новые языки программирования.

Наконец, все системы программирования в ограничениях различаются уровнем своей специализации.

Ниже мы попробуем указать место в этой классификации для ряда известных систем программирования в ограничениях.

5.1. "Чистое" программирование в ограничениях

5.1.1. UniCalc, Newton и Numerica

Одним из самых плодотворных примеров приложения программирования в ограничениях является решение численных задач. Давно разрабатываются системы для решения таких задач на основе вычислительных методов, известных математикам, однако не так давно они получили серьезных конкурентов в виде специализированных систем для решения численных задач на основе технологий, разработанных в рамках программирования в ограничениях. Наиболее известные системы из этого ряда — UniCalc [71, 72], Newton [34] и его развитие — система Numerica [73]. Все эти системы примерно одинаковы по своим возможностям. Они предлагают средства для спецификации задач удовлетворения ограничений над числовыми областями (целыми и вещественными). Для спецификации ограничений используются традиционные арифметические отношения и операции. Язык спецификации, по сути, совпадает с языком математических формул. Имеются специальные конструкции для задания дифференциальных уравнений и для спецификации задач математического программирования. Значения переменных представляются интервалами. Основным алгоритмом удовлетворения ограничений является интервальный аналог алгоритма достижения локальной совместности AC-3, в комбинации с ним решатели используют ряд специализированных алгоритмов: метод Гаусса, базисы Грёбнера, метод Ньютона и др.

5.1.2. ILOG Solver и системы на его основе

ILOG Solver реализован в виде библиотеки классов языка программирования Си++, что позволяет проводить его легкую интеграцию с другими системами программирования. Эта коммерческая система работает на нескольких платформах и может комплектоваться рядом вспомогательных продуктов, позволяющих повысить уровень интерфейса с системой (ILOG View) и расширить круг решаемых задач (ILOG Planner и ILOG Scheduler) [74]. Наконец, следует отметить, что ILOG Solver имеет на сегодняшний день лучшие технические характеристики, чем его конкуренты [75]. Что касается областей значений переменных, то они могут быть целыми, вещественными, логическими, множествами. Используются все стандартные операции и отношения над этими областями. Список областей может быть расширен пользователем, если последний искушен в Си++. Способ записи ограничений совпадает с возможностями записи выражений языка Си++. Значения переменных представляются в виде

интервалов. В качестве алгоритма используется модификация AC-5 [24]. Кроме того, в систему интегрировано множество специализированных алгоритмов для решения задач из разных областей.

Однако при всех своих плюсах ILOG Solver обладает, по нашему мнению, следующими недостатками.

- Использование в качестве языка спецификации задачи Си++ выглядит неоправданным, так как исключает возможность использования Solver пользователем-непрограммистом.
- Использование динамического языка программирования для спецификации статических задач затрудняет описание семантики системы. Не случайно сами авторы предпочитают относить ILOG Solver к языкам логического программирования в ограничениях [75], что противоречит сложившемуся к настоящему моменту пониманию таких языков [76, 77] (в частности, в ILOG Solver отсутствует понятие *унификации*).

5.1.3. Офисные системы

Перспективность коммерческого приложения систем программирования в ограничениях впервые была осознана в Российском научно-исследовательском институте искусственного интеллекта и в предшествующих ему организациях (Государственная научная фирма "Интеллектуальная технология"и лаборатория искусственного интеллекта ВЦ СО АН СССР). Такие системы, рассчитанные на конечного пользователя-непрограммиста, появились здесь еще во второй половине 80-х годов. Одной из первых таких систем был решатель UniCalc, рассмотренный выше. Конечно, самыми популярными программами были и будут оставаться в ближайшее время так называемые *офисные программы*, к которым относят текстовые редакторы, системы управления базами данных, электронные таблицы и системы календарного планирования. Система Time-EX [15] является офисной программой календарного планирования и удовлетворяет всем формальным требованиям, предъявляемым к пользовательским интерфейсам систем планирования. Однако использование в этой системе технологии программирования в ограничениях на основе аппарата недоопределенных моделей позволяет поднять ее функциональные возможности на качественно новый уровень. Благодаря недоопределенности значений параметров плана, последний может динамически сопровождаться в течение всего срока выполнения, а информация может уточняться как пользователем, так и системой.

Другой областью приложения технологии удовлетворения ограничений являются электронные таблицы. Система электронных таблиц ФинПлан [78] позволяет работать с интервальными параметрами. Зна-

чения в ячейках таблицы не разделяются на входные и выходные (параметры и формулы), более того, для одной ячейки может быть задано несколько формул (ограничений), которым значение в данной ячейке должно удовлетворять. Тем не менее электронная таблица — это не просто удобный способ спецификации задачи удовлетворения ограничений, это новый уровень взаимодействия с пользователем. Так как пользователь теперь имеет возможность изменять значение в любой ячейке таблицы, он может решать как прямые, так и обратные задачи для своей модели. Наличие такой возможности позволяет менеджеру принимать более взвешенные стратегические решения, чем он мог бы это сделать с помощью обычных таблиц.

Близкая по возможностям к ФинПлан система, расширяющая функциональность коммерческой электронной таблицы Microsoft Excel, описана в работе [79].

В работе [80] обсуждаются перспективы использования недоопределенности в пользовательских интерфейсах. Предлагается новый вид графического интерфейса — *активные диаграммы*, которые позволяют поднять взаимодействие системы с пользователем на новый уровень. Показано, что рассмотрение исследуемой пользователем вычислительной модели как задачи удовлетворения ограничений и применение к ней методов достижения локальной совместности (на основе аппарата недоопределенных моделей) приносит качественно новые свойства пользовательскому интерфейсу.

5.1.4. Система НеМо+

Система НеМо+ [1–3] является совместной разработкой Российского научно-исследовательского института искусственного интеллекта и Института систем информатики им. А. П. Ершова СО РАН и относится к классу конечно-пользовательских систем, предлагающих свой собственный язык представления знаний. Предметная область задачи описывается в терминах классов и отношений с использованием объектно-ориентированного подхода. Класс — основная единица языка НеМо+ — инкапсулирует в себе как область значений объектов этого класса, так и ограничения, наложенные на значения объектов и составляющих их подобъектов (слотов). Еще одной отличительной особенностью языка представления знаний является его модульность. Кроме того, система НеМо+ имеет отдельный вход на уровне инструментального языка программирования Си++, где пользователь может расширять набор базовых классов и отношений (т. е. тех классов и отношений, которые не могут быть выражены через уже существующие).

Набор областей значений переменных включает в себя целые и веще-

ственные числа, логические значения, строки, произвольные конечные области, заданные перечислением элементов, составные значения (записи и массивы), а также множества значений любого из этих типов. Операции и отношения над областями значения включают в себя традиционный набор арифметических, строковых, логических, теоретико-множественных операций, так же как и операций над составными объектами (конструкторы и селекторы). Допускается возможность задания экстенсиональных отношений (в виде таблиц). Ограничения записываются на языке термов, составленных из имен объектов, констант, операций и функций по обычным правилам. В языке допустимы условные ограничения, конструкторы альтернатив и универсальные ограниченные кванторы (декларативный способ задания циклических конструкций). Значения переменных могут представляться четырьмя стандартными способами: точное значение, перечисление значений, интервал значений и мультиинтервал значений. Язык спецификации недоопределенных значений может быть расширен любым другим видом недоопределенности.

В настоящее время в системе НеМо+ реализован только один универсальный алгоритм удовлетворения ограничений — обобщение алгоритма AC-3, которое восходит к предложенному А. С. Нариньяни алгоритму вычислений в недоопределенных моделях [7]. Кроме того, в системе реализованы средства поиска точных и оптимальных решений задачи: порождение альтернативных ветвей и откаты.

НеМо+ является идеологическим наследником системы НеМо-ТеК [81] — технологического комплекса для спецификации и решения задач на основе аппарата недоопределенных моделей. Комплекс НеМо+ отличается от НеМо-ТеК более развитым языком представления знаний, иерархическим способом организации вычислительной сети, расширяемой архитектурой.

5.2. Логическое программирование в ограничениях

Декларативный характер спецификации задачи удовлетворения ограничений привел многих исследователей к мысли, что наиболее просто встроить средства спецификации таких задач в какой-либо существующий декларативный язык. Наиболее известный из декларативных языков, разработанных в ИИ, — это Пролог [82]. Схема интеграции логического программирования в языке Пролог с программированием в ограничениях получила название CLP (Constraint Logic Programming — логическое программирование в ограничениях). Языки CLP имеют ясную семантику, описанию которой посвящено много работ (наиболее известные из них — [76, 77]). Рассмотрим известные языки этого класса.

5.2.1. CLP($\mathcal{R}$) и Пролог III

CLP($\mathcal{R}$) [36] и Пролог III [83] — исторически первые языки логического программирования в ограничениях. В этих системах язык термов Пролога расширяется языком спецификации ограничений над вещественными числами. Алгоритмы удовлетворения ограничений основаны на распространении точных значений, при этом используются специализированные алгоритмы для решения специальных подсистем задачи (таких, как системы линейных уравнений). Затем результаты алгоритмов подставляются в исходную задачу, в результате чего она упрощается, в ней снова выделяются простые подзадачи и т. д. В качестве областей значений переменных используются рациональные числа (Пролог III) или числа с плавающей точкой (CLP($\mathcal{R}$)). В системе Пролог III допускаются также ограничения над списками с операцией конкатенации [83].

5.2.2. CHIP и clp(FD)

CHIP ([84], название расшифровывается как Constraint Handling In Prolog) — первый язык логического программирования, в котором реализован алгоритм удовлетворения ограничений AC-3 [6] над конечными областями значений. Успех этого языка программирования [84] послужил мощным стимулом к развитию программирования в ограничениях и созданию новых систем на базе этого подхода.

В языке clp(FD) [85] предпринята попытка встроить алгоритм удовлетворения ограничений над конечными областями в Пролог другим способом. Он заключается в использовании одного базового ограничения "значение переменной принадлежит заданному множеству значений", на основе которого в языке по обычным правилам Пролога определяются все ограничения более высокого уровня. Эффективный транслятор с этого языка в язык Си позволил получить на стандартных тестах более сильные результаты, чем в CHIP [85]. До появления ILOG Solver [75] система clp(FD) оставалась самым эффективным средством решения задач удовлетворения ограничений над конечными областями.

5.2.3. CLP(BNR)

Система CLP(BNR) [47] — первая система логического программирования в ограничениях, в которой были полностью реализованы средства спецификации ограничений над вещественными переменными и решение задач на основе удовлетворения ограничений над интервальными значениями. Кроме вещественных областей, язык поддерживает целочис-

ленные и логические переменные и ограничения над ними. Несмотря на недостаточную эффективность, CLP(BNR) послужила прообразом для других подобных систем [34].

5.2.4. Flang

Язык Flang [86] основан на интеграции сразу трех парадигм программирования: логического, функционального и программирования в ограничениях. Поддерживая удовлетворение ограничений над интервальными значениями, Flang может решать задачи над целыми и вещественными областями. От функционального программирования язык берет мощный набор средств для спецификации ограничений. Кроме того, в схему решения задач входят специальные средства обработки дизъюнктивных ограничений.

5.2.5. Conjuncto и CLP(Σ^*)

Применение алгоритмов удовлетворения ограничений к областям-множествам рассматривалось в работах [40] и [42]. Этими авторами предложены языки логического программирования с возможностью спецификации ограничений над множествами Conjuncto и CLP(Σ^*). Стоит отметить, что А. С. Нариньяни, наоборот (и гораздо ранее), сначала предложил понятие недоопределенности для множеств [11], а затем расширил его на другие (в том числе числовые) области [7]. В качестве областей значений переменных в таких задачах выступает множество всех подмножеств некоторого множества, а в качестве отношений — традиционные теоретико-множественные отношения и операции. Семантика интервала над множествами рассмотрена подробно в работе [51]. Отметим, что ограничения над множествами используются и в ряде других систем: Яхно и Петров [43], Puget [41], а также в рассмотренной выше системе НеМо+ (Телерман и др. [1], Г. Загорулько и др. [3], Швецов и др. [2]). В эти системы встроены средства спецификации и решения задач удовлетворения ограничений над множествами.

5.3. Императивное программирование в ограничениях

Императивные языки программирования реже служили платформой для интеграции в них методов спецификации и решения задач удовлетворения ограничений. Причина тому — отсутствие в императивных языках декларативных конструкций, с помощью которых можно задавать ограничения на значения переменных. В этой области, как правило, разра-

батываются новые языки представления и обработки знаний на основе интеграции императивного подхода и программирования в ограничениях. Рассмотрим известные системы этого класса.[1]

5.3.1. Kaleidoscope'91 и TAO

Язык Kaleidoscope'91 [17] является обычным императивным объектно-ориентированным языком программирования со встроенными средствами для спецификации ограничений над переменными программы. Классы в этом языке описывают структуру группы объектов и ограничения, наложенные на их значения. Если значение какого-либо объекта изменяется, автоматически удовлетворяются все ограничения, в которые он входит. Алгоритм не пытается достичь совместности системы ограничений: просто новые значения вычисляются по старым (реактивное распространение ограничений). Приоритет удовлетворения ограничений задается пользователем путем спецификации *иерархии ограничений* [57]. Таким образом, ограничения в данном языке служат для задания дополнительного потока управления программы. Конечно, в этой схеме переменные могут иметь только точные значения, зато допускаются любые типы данных и отношения над ними.

Система TAO [18, 87] интегрирует в себе следующие свойства:

- объектно-ориентированное программирование в ограничениях,
- асинхронное и последовательное управление,
- вычисление с частично известной информацией.

Две основные составляющие системы TAO — это (активные) объекты (или *агенты*) и время. Время определяет последовательность тактов. На каждом такте объекты получают сообщения (они могут приходить как от других объектов, так и извне системы), которые они обрабатывают. Каждый объект имеет ограничения, наложенные на значения его компонентов, которые определяют внутреннюю задачу удовлетворения ограничений для объекта; эти ограничения также автоматически удовлетворяются на каждом такте. Для удовлетворения ограничений используется не иерархическое распространение точных значений, как в Kaleidoscope'91, а метод достижения локальной совместности на основе недоопределенных моделей.

[1] Мы относим систему ILOG Solver к классу "чистого" программирования в ограничениях, хотя при спецификации задачи удовлетворения ограничений в ней можно пользоваться такими императивными конструкциями языка Си++, как условия и циклы. Однако возможность перехода из одного состояния задачи в другое в этой системе весьма затруднительна, что служит, на наш взгляд, веской причиной не относить ILOG Solver к императивному программированию в ограничениях.

Обе системы могут применяться в таких областях, как разработка пользовательских интерфейсов, систем реального времени и др.

5.3.2. Параллельное программирование в ограничениях

Другой подход к императивному программированию в ограничениях состоит в использовании *constraint store* — выделенного объекта программы, играющего роль машины логического вывода. К этому объекту применимы две операции: `tell` и `ask`. Первая сообщает объекту новую информацию, вторая запрашивает истинность или ложность некоторой информации. Вся информация представляется в виде ограничений, вывод новой информации происходит с помощью алгоритмов удовлетворения ограничений. Выделяются три вида управляющих конструкций: последовательное выполнение, параллельное выполнение и недетерминированный выбор. Такая схема вычислений получила название *параллельного программирования в ограничениях* и была предложена в работе Saraswat [88].

5.4. Активные базы данных

Еще одной областью приложения программирования в ограничениях послужили системы управления базами данных (СУБД). Здесь различают два основных подхода:

- использование методов удовлетворения ограничений для эффективного выполнения запросов к БД [89];
- изменение формата представления данных в БД с учетом недоопределенности значений. Этот подход приводит к понятию *обобщенного кортежа*, предложенного в [90]. Для работы с такими данными используются традиционные методы удовлетворения ограничений.

Как уже отмечалось, применение этих подходов имеет важное значение в областях ГИС и САПР. Одной из первых разработок в данной области является объектно-ориентированная СУБД с ограничениями C^3 [91]. Объекты в этой базе данных могут содержать данные трех типов: пространственные, временные и ограничения. Все манипуляции с такими объектами осуществляются посредством *исчисления ограничений*.

Отметим также, что в системе НеМо+ имеется возможность задания отношений над объектами с помощью таблиц, формирующихся в результате SQL-запроса к обычной реляционной базе данных [92].

6. ЗАКЛЮЧЕНИЕ

Как следует из приведенного обзора, парадигма программирования в ограничениях является весьма популярной, в ее рамках реализуются новые языки и системы программирования, исследуется их семантика, создаются приложения для решения различных промышленных задач.

В таблице перечислены известные системы программирования в ограничениях вместе с их классификацией по различным параметрам. Данная таблица суммирует материал, изложенный в последнем разделе настоящей работы. В первом столбце дается название системы, во втором перечисляются области значений переменных, с которыми работает система, в третьем — способ представления значений переменной, в четвертом — язык описания ограничений, в пятом — используемые для решения алгоритмы, в шестом — парадигма программирования, в рамках которой разработана система (CP — "чистое" программирование в ограничениях, CLP — логическое программирование в ограничениях, ICP — императивное программирование в ограничениях). Мы не приводим в этой таблице классификацию по типу используемых отношений, поскольку из рассматриваемых систем только НеМо+ обладает возможностью спецификации экстенсиональных и интенсиональных отношений одновременно. Все остальные системы используют интенсиональные отношения.

Из представленной классификации видно, что в области программирования в ограничениях самый общий подход реализован в рамках системы НеМо+. Система обладает выразительным языком спецификации знаний в виде задач удовлетворения ограничений, для решения которых используется универсальный алгоритм удовлетворения ограничений, работающий над произвольными областями. По спектру решаемых задач система превосходит большинство других систем программирования в ограничениях. В настоящее время ведутся активные работы по увеличению эффективности системы НеМо+ для ее промышленного применения в решении задач из области САПР.

БЛАГОДАРНОСТИ

Авторы выражают признательность И. В. Поттосину, предложившему идею настоящей статьи и высказавшему ряд ценных замечаний по ее улучшению, а также А. С. Нариньяни, И. Е. Швецову и Т. П. Кашеваровой, которые любезно прочитали работу и указали на ее недостатки.

Классификация систем программирования в ограничениях

Система	Области	Значения	Ограничения	Алгоритмы	Схема
НеМо+	Конечные, числовые, составные, определяемые пользователем (классы)	Точные, перечислимые, интервальные, мультиинтервальные, определяемые пользователем	Расширяемый язык термов	AC-3, порождение альтернатив и откаты, методы поиска оптимального решения	CP
Пролог III	Числа с плавающей точкой, списки	Точные	Язык термов	Комбинация специализированных алгоритмов	CLP
ТАО	Целые, вещественные, классы	Интервалы	Язык термов	AC-3, реактивное распространение ограничений	ICP
ФинПлан	Целые и вещественные	Интервалы	Язык термов	AC-3	CP
CHIP	Конечные	Перечисления	Примитивные	AC-3	CLP
CLP(BNR)	Целые и вещественные	Интервалы	Расширяемый язык термов	AC-3	CLP
clp(FD)	Конечные	Интервалы и перечисления	Примитивные	AC-3	CLP
CLP($\mathcal{R}$)	Рациональные числа	Точные	Язык термов	Симплекс-метод, задержанные вычисления	CLP
CLP(Σ^*)	Конечные множества	Интервалы	Примитивные	AC-3	CLP
Conjuncto	Конечные множества	Интервалы	Примитивные	AC-3	CLP
Flang	Целые и вещественные	Интервалы	Расширяемый язык термов	AC-3, обработка дизъюнкций	CLP

Окончание таблицы

Система	Области	Значения	Ограничения	Алгоритмы	Схема
ILOG Solver	Целые, вещественные, булевы, конечные множества	Интервалы	Язык термов Си++	AC-5, множество специализированных алгоритмов	CP
Kaleidoscope	Целые, вещественные, классы	Точные	Язык термов	Иерархическое реактивное распространение ограничений	ICP
Newton	Вещественные	Интервалы	Язык термов	AC-3, метод Ньютона	CLP
Numerica	Вещественные	Интервалы	Язык термов	AC-5, численные методы	CP
Time-EX	Временные интервалы (работы)	Интервалы	Примитивные	AC-3	CP
UniCalc	Целые, вещественные, булевы	Интервалы	Язык термов	AC-3, символьные преобразования, поиск точного решения	CP

СПИСОК ЛИТЕРАТУРЫ

1. **Telerman V., Sidorov V., Ushakov D.** Problem Solving in the Object-Oriented Technological Environment NeMo+ // Proc. of the 2nd Intern. A. P.Ershov Memorial Conf. "Perspectives of System Informatics". — Berlin a.o.: Springer-Verlag, 1996. — P. 91–100. — (Lect. Notes in Computer Sci.; Vol. 1181).
2. **Shvetsov I., Telerman V., Ushakov D.** NeMo+: Object-Oriented Constraint Programming Environment Based on Subdefinite Models // Proc. of CP'97. "Principles and Practice of Constraint Program". — Springer-Verlag, 1997. — P. 534–548. — (Lect. Notes in Comput. Sci.; Vol. 1330).
3. **Загорулько Г., Сидоров В., Телерман В. и др.** Обстановка для программирования в ограничениях на основе недоопределенных моделей NeMo+ (язык, архитектура, интерфейс) // Научно-техн. отчет N 7 / Российский НИИ искусственного интеллекта, Институт систем информатики им. А. П. Ершова СО РАН. — Москва; Новосибирск, 1998. — 107 с.
4. **Тыугу Э. Х.** Концептуальное программирование. — М.: Наука, 1984. — 255 с.

5. **Benhamou F.** Heterogeneous constraint solving // Proc. of the 5th Intern. Conf. on Algebraic and Logic Program. (ALP 96). — Berlin a.o.: Springer-Verlag, 1996. — P. 62–76. — (Lecture Notes in Computer Sci. — Vol. 1139).
6. **Mackworth A. K.** Consistency in Networks of Relations // Artificial Intelligence. — 1977. — N 8. — P. 99–118.
7. **Нариньяни А. С.** Недоопределенные модели и операции с недоопределенными значениями. — Новосибирск, 1982. — 33 с. — Препр./АН СССР. Сиб. отд-ние. ВЦ; N 400.
8. **Mohr R., Henderson T.** Arc-consistency and path-consistency revisited // Artificial Intelligence. — 1986. — N 28. — P. 225–233.
9. **Kumar V.** Algorithms for Constraint Satisfaction Problems: A Survey // AI Magazine. — 1992. — Vol.13(1). — P. 32–44.
10. **Нариньяни А. С.** Искусственный интеллект: стагнация или новая перспектива // КИИ'98. Шестая национальная конф. с междунар. участием: Сб. науч. трудов: В 3 т. — Т.1. — Пущино, 1998. — С. 15–29.
11. **Нариньяни А. С.** Недоопределенные множества — новый тип данных для представления знаний. — Новосибирск, 1980. — 33 с. — (Препр./АН СССР. Сиб. отд-ние. ВЦ; N 232).
12. **Нариньяни А. С.** Недоопределенность в системах представления и обработки знаний // Изв. АН СССР. Сер. "Техн. кибернетика." — 1986. — N 5. — С. 3–28.
13. **Shvetsov I., Semenov A., Telerman V.** Application of Subdefinite Models in Engineering // Artificial Intelligence in Engineenring. — 1997. — N 11. — P. 15–24.
14. **Лихачев Д. Б., Яхно Т.М.** Использование обобщенных вычислительных моделей для авоматизации проектирования // Информатика: инструментальные средства. — Новосибирск, 1988. — С. 53–71.
15. **Нариньяни А. С., Седреева Г. О., Седреев С. В., Фролов С. А.** TimeEX/Windows — новое поколение недоопределенной технологии календарного планирования // Проблемы представления и обработки не полностью определенных знаний/ Под ред. И. Е. Швецова. — Москва; Новосибирск, 1996. — С. 101–116.
16. **Waltz D.** Understanding line drawing in scenes with shadows // The Psychology of Computer Vision. — McGraw-Hill, 1975. — P. 19–91.
17. **Freeman-Benson B. N., Borning A.** Integrating Constraints with an Object-Oriented Language // Proc. of ECOOP'92. — 1992. — P. 268–286.
18. **Швецов И. Е.** Основные положения технологии активных объектов. — Новосибирск, 1995. — 26 с. — (Препр./НФ РосНИИ ИИ).
19. **Sutherland I.** Sketchpad: A Man-Machine Graphical Communication System: PhD Thesis. — Department of Electrical Engineering, MIT, 1963.
20. **Тыугу Э. Х.** Решение задач на вычислительных моделях // ЖВМиМФ. — 1970. — Т. 10, N 5.
21. **Montanari U.** Networks of Constraints: Fundamental Properties and Application to Picture Processing // Information Sci. — 1974. — N 7(2). — P. 95–132.
22. **Burstall R. M.** A program for solving word sum puzzles // Comp. J. — 1969. — N 12. — P. 48–51.
23. **Fikes R. E.** REF-ARF: A System for Solving Problems Stated as Procedures // Artificial Intelligence. — 1970. — N 1(1). — P. 27–120.
24. **Deville Y., Van Hentenryck P.** An Efficient Arc Consistency Algorithm for a Class of CSP Problems // Proc. IJCAI. — 1991. — P. 325–330.
25. **Bessière C.** Arc-consistency and arc-consistency again // Artificial Intelligence. — 1994. — N 65. — P. 179–190.
26. **Besière C., Freuder E. C., Regin J.-C.** Using inference to reduce arc consistency computation // Proc. 14th IJCAI. — 1996. — Vol. 1. — P. 592–598.
27. **Телерман В. В.** Применение обобщенных вычислительных моделей для реали-

зации вывода/вычислений в базах знаний // Тез. докл. всесоюз. конф. "Проблемы развития и освоения интеллектуальных систем". Секция II: Методы и модели освоения интеллектуальных систем. — Новосибирск, 1986. — С. 80–81.
28. **Шокин Ю. И.** Интервальный анализ. — Новосибирск: Наука, 1981. — 112 с.
29. **Davis E.** Constraint Propagation with Interval Labels // Artificial Intelligence. — 1987. — N 32. — P. 281–331.
30. **Cleary J. G.** Logical Arithmetic // Future Gener. Comp. Systems. — 1987. — Vol. 2(2). — P. 125–149.
31. **Hyvönen E.** Constraint Reasoning Based on Interval Arithmetic // Proc. IJCAI. — 1989. — P. 193–199.
32. **Older W., Vellino A.** Extending Prolog with Constraint Arithmetic on Real Intervals // Proc. Canad. Conf. on Electrical and Computer Engineering. — 1990.
33. **Lhomme O.** Consistency Techniques for numeric CSPs // Proc. IJCAI'93. — Chambery, France, 1993. — P. 232–238.
34. **Benhamou F., McAllester D., Van Hentenryck P.** CLP(Intervals) Revisited // Proc. 1994 Intern. Logic Progr. Symp. — MIT Press, 1994. — P. 124–138.
35. **Shary S. P.** A New Approach to the Analysis of Static Systems Under Interval Uncertaity // Scientific Computing and Validated Numerics. — Berlin: Akademie Verlag, 1996. — P. 118–132.
36. **Jaffar J., Michayov S., Stuckey P. J., Yap R. H. C.** The CLP($\mathcal{R}$) Language and System // TOPLAS: ACM Transactions Progr. Languages and Systems. — 1992. — Vol. 14(3). — P. 339–395.
37. **Freuder E. C.** Synthesizing Constraint Expressions // Com. of the ACM 21. — 1978. — N 11. — P. 958–966.
38. **Freuder E. C., Wallace R. J.** Partial Constraint Satisfaction // Artificial Intelligence. — 1992. — N 58. — P. 21–70.
39. **Freuder E.C.** Exploiting Structure in Constraint Satisfaction Problems// Constraint Progr.: Proc. NATO ASI Parnu, Estonia. — Springer-Verlag, 1994. — P. 54–79.
40. **Walinsky C.** CLP(Σ^*): Constraint Logic Programming with Regular Sets // ICLP'89: Proc. 6th Intern. Conf. on Logic Progr. — MIT Press, 1989. — P. 181–196.
41. **Puget J.-F.** PECOS A High Level Constraint programming Language // Proc. of Spicis 92. — Singapore, Sept., 1992.
42. **Gervet C.** Conjuncto: Constraint Logic Programming with Finite Set Domains // Technical report of ECRC, EC-94-15, 1994.
43. **Yakhno T., Petrov E.** LogiCalc: Integrating Constraint Programming and Subdefinite Models // Proc. PACT'96. — 1996. — P. 357–372.
44. **Telerman V., Ushakov D.** Subdefinite Models as a Variety of Constraint Programming // Proc. 8th Intern. Conf. on Tools with Artificial Intelligence. ICTAI'96. — IEEE Computer Soc., 1996. — P. 157–163.
45. **Caseau Y., Puget J.-F.** Constraints on Order-Sorted Domains // Proc. ECAI'94 Workshop on Constraint Processing. — Amsterdam, 1994.
46. **van Emden M. H.** Value constraints in the CLP scheme // Constraints. — 1997. — Vol. 2(2). — P. 163–184.
47. **Benhamou F., Older W. J.** Applying Interval Arithmetic to Real, Integer and Boolean Constraints // J. Logic Progr. — 1997. — Vol. 32(1). — P. 1–24.
48. **Telerman V., Ushakov D.** Data Types in Subdefinite Models // Proc. of Intern. Conf. Artificial Intelligence and Symbolic Mathematical Computation (AISMC-3). — Berlin a.o.: Springer-Verlag, 1996. — P. 305–319. — (Lect. Notes in Comp. Sci.; Vol. 1138).
49. **Cousot P., Cousot R.** Abstract Interpretation: A Unified Lattica Model for Static Analysis of Programs by Constructions or Approximation of Fixpoints // Proc. 4th

ACM Symp. of Principles of Programming Languages. — 1977.
50. **Apt K. R.** The Essence of Constraint Propagation // J. Logic Progr., to appear.
51. **Ushakov D.** Some Formal Aspects of Subdefinite Models. — Novosibirsk, 1998. — 23 p. — (Prepr./ A.P. Ershov Institute of Informatics Systems, Siberian Division of Russian Academy of Sciences; N 49).
52. **Benhamou F.** Interval Constraint Logic Programming // Chatillon Spring School "Constraint Programming: Basics and Trends". Selected Papers. — Berlin a.o.: Springer-Verlag, 1995. — P. 1–21. — (Lect. Notes in Comp. Sci.; Vol. 910).
53. **Mayoh B.** Constraint Programming and Artificial Intelligence // Constraint Programming: Proc. 1993 NATO ASI Parnu, Estonia. — Springer-Verlag, 1993. — P. 18–53.
54. **Van Hentenryck P., Saraswat V., et al.** Strategic Directions in Constraint Programming // ACM Computing Surveys. — 1996. — Vol. 28, N 4.
55. **Яхно Т. М.** Программирование в ограничениях: обзор и классификация подходов и методов // Системная информатика, вып. 4. — Новосибирск: Наука, 1995. — С. 160–192.
56. **Нильсон Н.** Принципы искусственного интеллекта. — М.: Радио и связь, 1985.
57. **Borning A., Freeman-Benson B., Wilson M.** Constraint Hierarchies // Constraint Programming: Proc. 1993 NATO ASI Parnu, Estonia. — Springer-Verlag, 1994. — P. 80–122.
58. **Shokin Y. I.** On Interval Problems, Interval Algorithms and Their Computational Complexity // Scientific Computing and Validated Numerics. — Berlin: Akademie Verlag, 1996. — P. 314–328.
59. **Алефельд Г., Херцбергер Ю.** Введение в интервальные вычисления. — М.: Мир, 1987. — 260 с.
60. **Яковлев А. Г.** Машинная арифметика мультиинтервалов // Вопросы кибернетики. Проблемно-ориентированные вычислительные системы. — 1987. — С. 66–81.
61. **Телерман В. В.** Использование мультиинтервалов в недоопределенных моделях // Тез. докл. X всесоюзю семинара "Параллельное программирование и высокопроизводительные системы: Методы представления знаний в информационных технологиях". — Киев, 1990.
62. **Cleary J. G.** Constructive Negation of Arithmetic Constraints // Proc. of Workshop on Interval Constraints, ILPS'95. — Portlend, Oregon, USA, 1995.
63. **Gaschnig J.** A General Backtracking Algorithm that Eliminates Most Redundant Tests // Proceeding IJCAI-77. — 1997. — P. 457.
64. **Gaschnig J.** Performance Measurement and Analysis of Certain Search Algorithms // Technical Report CMU-CS-79-124. — Carnegie-Mellon University. — Pittsburgh, PA, 1979.
65. **Haralick R. M., Elliott G. L.** Increasing Tree Search Efficiency for Constraint Satisfaction Problems // Artificial Intelligence. — 1980. — N 14. — P. 263–313.
66. **Ginsberg M. L.** Dynamic Backtracking // J. Artificial Intelligence Research. — 1993. — N 1. — P. 25–46.
67. **Sabin D., Freuder E.** Contradicting Conventional Wisdom in Constraint Satisfaction // Proc. of the 2nd Intern. Workshop "Principles and Practice of Constraint Program". — Berlin a.o.: Springer-Verlag, May 1994. — (Lect. Notes in Comp. Sci.; Vol. 874).
68. **Карманов В. Г.** Математическое программирование. — М.: Наука, 1986.
69. **Semenov A., Kashevarova T., Leshchenko A., Petunin D.** Combining Various Techniques with the Algorithm of Subdefinite Calculations // Proc. 3rd Intern. Conf. on the Practical Application of Constraint Technology PACT'97. — London, England, April 1997. — P. 287–306.
70. **Загорулько Ю. А., Попов И. Г.** Представление знаний в интегрированной

технологической среде SemP-TAO // Проблемы представления и обработки не полностью определенных знаний/ Под ред. И. Е. Швецова. — Москва; Новосибирск, 1996. — С. 59–74.

71. **Babichev A. B., Kadyrova O. B., Kashevarova T. P., Leshchenko A. S., Semenov A. L.** UniCalc, A Novel Approach to Solving Systems of Algebraic Equations //Proc. Intern. Conf. on Numerical Analysis with Automatic Result Verifications. Lafayette, Louisiana, USA, 1993. — Interval Computations N 2. — 1993. — P. 29–47.
72. **Semenov A., Babichev A., Leshchenko A.** Subdefinite Computations and Symbolic Transformations in the UniCalc Solver // Proc. of the 2nd Intern. Conf. AISMC-2 "Integrating Symbolic Mathematical Computation and Artificial Intelligence". — Berlin a.o.: Springer-Verlag, 1995. — P. 264–275. — (Lect. Notes in Comp. Sci.; Vol. 958).
73. **Van Hentenryck P., Michel L., Deville Y.** Numerica: a Modeling Language for Global Optimization. — The MIT Press, Cambridge, Mass., 1997.
74. **ILOG Optimization Suite white paper.** `http://www.ilog.com/papers/optimization/`
75. **Puget J.-F., Leconte M.** Beyond the Glass Box: Constraints as objects // Logic Programming. Proc. 1995 Intern. Symp. — MIT Press, 1995. — P. 513–527.
76. **Van Hentenryck P.** Constraint Satisfaction Using Constraint Logic Programming // Artificial Intelligence. — 1992. — N 58. — P. 113–159.
77. **Jaffar J., Maher M. J.** Constraint Logic Programming: A Survey // J. Logic Progr. — 1994. — Vol. 19(20). — P. 503–581.
78. **Shvetsov I., Kornienko V., Preis S.** Interval spreadsheet for problems of financial planning // Proc. of PACT'96.
79. **Hyvönen E., De Pascale S.** Interval Computations on the Spreadsheet // Applications of Interval Computations. — Kluwer Academic Publishers, 1996. — P. 169–210.
80. **Левин Д. Я.** Активные диаграммы — новые пользовательские возможности на основе недоопределенных вычислений // Проблемы представления и обработки не полностью определенных знаний/ Под ред. И. Е. Швецова. — Москва; Новосибирск, 1996. — С. 117–122.
81. **Телерман В. В., Дмитриев В. Е.** Технология программирования на основе недоопределенных моделей. — Новосибирск, 1995. — 38 с. — (Препр./РАН. Сиб. отд-ние. ИСИ; N 25).
82. **Колмероэ А., Кануи А., ван Канегем М.** Пролог — теоретические основы и современное развитие // Логическое программирование. — М.: Мир, 1988. — С. 27–133.
83. **Colmerauer A.** An introduction to Prolog III // Communications of the ACM. — 1990. — Vol. 33(7).
84. **Van Hentenryck P.** Constraint Satisfaction in Logic Programming // Logic Progr. Series, MIT Press. — Cambridge, MA, 1989.
85. **Diaz D., Codognet P.** A minimal extension of WAM for clp(FD) // Proc. 10th Intern. Conf. of Logic Progr. — 1993.
86. **Манцивода А. В.** Программирование в ограничениях на Флэнге // Системная информатика, вып. 4. — Новосибирск: Наука, 1995. — С. 118–159.
87. **Швецов И. Е., Нестеренко Т. В., Старовит С. А., Титова М. В.** Технология активных объектов: от концепции к реализации // Проблемы представления и обработки не полностью определенных знаний/ Под ред. И. Е. Швецова. — Москва; Новосибирск, 1996. — С. 88–100.
88. **Saraswat V. A., Rinard M.** Concurrent Constraint Programming // Proc. ACM Symp. on Principles of Progr. Languages. — 1990. — P. 232–245.
89. **Voronkov A.** Merging Relational Database Technology with Constraint Techno-

logy // Proc. of the 2nd Intern. A. P.Ershov Memorial Conf. "Perspectives of System Informatics". — Berlin a.o.: Springer-Verlag, 1996. — P. 409–419. — (Lect. Notes in Comp. Sci.; Vol. 1181).

90. **Gaede V., Wallace M.** An Informal Introduction to Constraint Database Systems // Lect. Notes in Comp. Sci. — Springer-Verlag, 1997. — Vol. 1191. — P. 7–52.
91. **Brodsky A., Segal V. E.** The C^3 Constraint Object-Oriented Database System: An Overview // 2nd Intern. Workshop on Constraint Database Systems (CDB 97). Proc. of Constraint Databases and Applications. — Berlin a.o.: Springer-Verlag, 1997. — (Lect. Notes in Comp. Sci.; Vol. 1191).
92. **Lipski S., Sidorov V., Telerman V., Ushakov D.** Database Processing in Constraint Programming Paradigm Based on Subdefinite Models // Joint Bulletin of NCC & IIS. Ser.: Comp. Sci. — 1999. — N 12. — P. 29–31.