Санкт-Петербургский государственный университет
информационных технологий, механики и оптики
Кафедра компьютерных технологий

С.Э. Вельдер

Применение методов снижения размерности к задачам верификации TCTL и оптимальной укладки графов

Магистерская диссертация

Научный руководитель – профессор А.А. Шалыто

Санкт-Петербург
2008

Содержание

Введение

Диссертация посвящена нескольким задачам, которые связаны между собой отношением сводимости.

Первая задача – это задача верификации *TCTL*-свойств на автоматах с таймерами [1–17]. В настоящей работе выполнен обзор существующих алгоритмов верификации указанных свойств, проанализированы их достоинства и недостатки и разработан алгоритм, соединяющий в себе их достоинства, который свободен от недостатков известных алгоритмов. Основной элемент алгоритма верификации – построение модели Крипке (регионного автомата) по автомату с таймерами и *TCTL*-формуле, а основное достоинство алгоритма, разработанного в данной диссертации – малый размер получившейся модели Крипке.

Поскольку задача верификации может быть эффективно сведена к задаче выполнимости (satisfiability problem, *SAT*), которая решается эвристически специальными программными средствами (*SAT*-solver'ами), возникла необходимость в исследовании задачи выполнимости на предмет поиска эвристики, которая была бы эффективнее уже существующих. Эффективность здесь понимается в смысле оценки временной сложности в худшем случае.

Для этого было выполнено сведение задачи *SAT* к комбинаторной задаче на игровом поле [18 – 41]. Существует широко распространённый метод

такого сведения: вначале по булевой формуле строится булева схема из конечного набора функциональных элементов, после этого данная схема «рисуется» на игровом поле рассматриваемой задачи: для каждого функционального элемента строится конфигурация, представляющая его в терминах игры. Построение такой конфигурации может проводиться как вручную, так и автоматически [41]. В данной работе вручную выполнено сведение *SAT* к задаче *1* - *MINESWP*. Причина выбора этой задачи состоит в том, что её игровое поле может иметь произвольную размерность (стандартная двумерная игра «Сапёр» легко обобщается на *k*-мерную).

Задачу выполнимости полезно свести к комбинаторной игре потому, что игра имеет алгоритм решения более эффективный, чем полный перебор – динамическое программирование по профилю.

Эффективность динамики по профилю зависит от объёма игрового поля, в который вложена рассматриваемая булева схема. Поскольку схема представляет собой граф специального вида, имеет смысл изучить асимптотические оценки на объём (многомерной) области, в которую может быть вложен такой граф. В работе получены оценки для такой величины.

Нижняя оценка худшего случая $\Omega\left(n^{\frac{1}{k-1}}\right)$, полученная в диссертации, выполняется почти для всех графов. В работе доказывается достаточное условие на граф, любая реализация которого в пространстве удовлетворяет

нижней оценке. Графы, удовлетворяющие этому условию, называются универсальными.

Наконец, получение нижней оценки позволяет задать вопрос: для какого числа графов (для какой доли множества всех графов) выполняется достаточное условие нижней оценки? И какова будет нижняя оценка не для худшего, а для среднего случая? В работе даётся следующий ответ на эти вопросы: доля таких графов стремится к единице с ростом числа вершин. Следовательно, оценка в худшем случае совпадает с оценкой в среднем случае.

Доказательство описанной сходимости выполняется путём предъявления конкретной формулы, выражающей точное число универсальных графов в зависимости от параметров.

Все оценки, полученные в работе, являются обобщениями существующих оценок, которые строились, главным образом, для плоских и трёхмерных вложений. Оценки для вложений графов в пространства размерности выше трёх ранее исследовались только одним автором (D. R. Wood) [72 – 74, 82]. Им были получены верхние оценки для вложения графа общего вида (в данной работе рассматриваются графы специального вида). Более подробно об этих оценках, а также их прикладных значениях изложено в работах [42 – 84].

Задача оптимального вложения графов сама по себе является актуальной и непростой математической проблемой. Ранее у неё было только два приложения: эстетическая визуализация графа (в этой области актуальны, как правило, плоские вложения) и построение компактным микросхем (исследуются трёхмерные вложения). Также у нижней оценки на реализацию графа в трёхмерном пространстве имелись биологические предпосылки [42]. Данная работа открывает (новую) связь задачи вложения графа в минимальный объём и теорией сложности алгоритмов.

Глава 1. Проверка моделей для асинхронных систем

1.1. Автоматы с таймерами

Автоматы с таймерами [1, 2] используются для верификации асинхронных систем. Они являются почти такими же объектами, как и модели Крипке, но с некоторыми отличиями: автомат с таймерами в течение всей работы сопровождают несколько переменных-таймеров, и для каждого состояния и перехода указано условие на таймеры, которое должно выполняться в этом состоянии или на этом переходе. Кроме того, при выполнении переходов некоторые заданные таймеры могут сбрасываться.

Рассмотрим в качества примера автоматы с одним таймером и возможные исполняющие сценарии для них (рис. 1 – рис. 3). Слева на рисунках указаны автоматы, справа – возможные варианты работы таймеров.

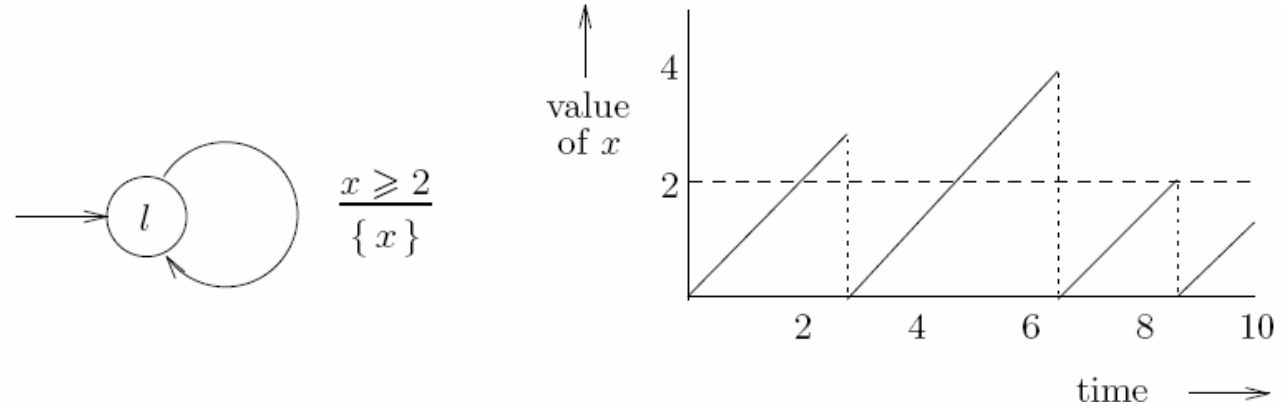


Рис. 1. Пример простого автомата с одним таймером

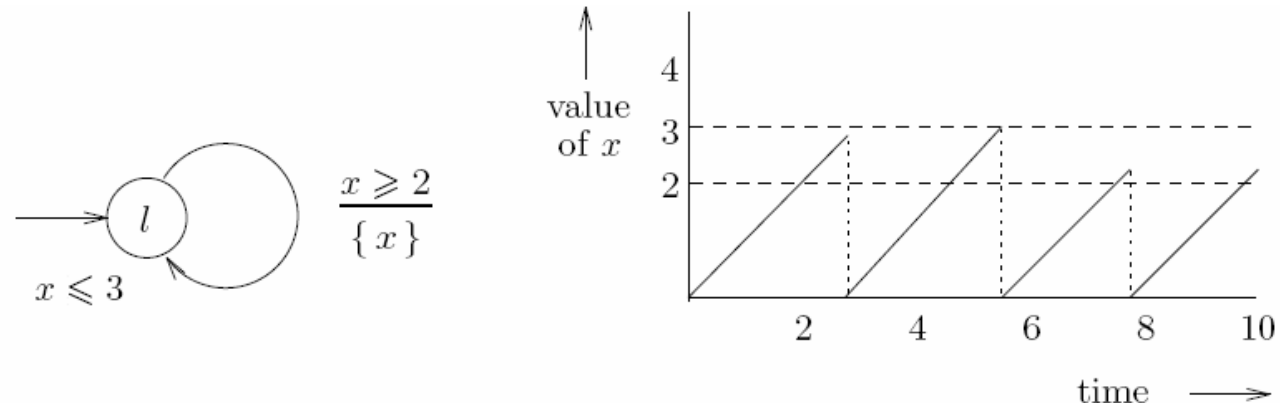


Рис. 2. Добавим инвариант на состояние

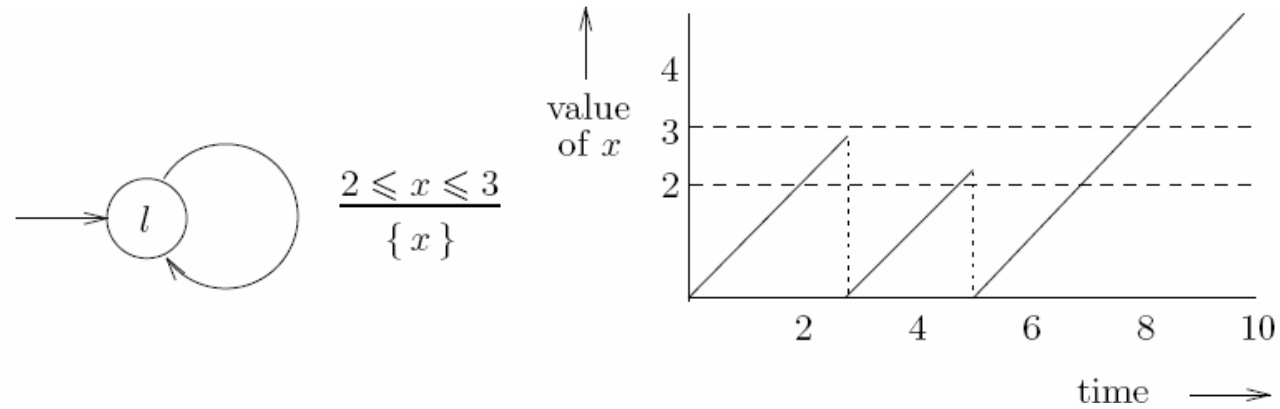


Рис. 3. Видоизменим ограничения

Приведем теперь пример автомата с двумя таймерами (рис. 4).

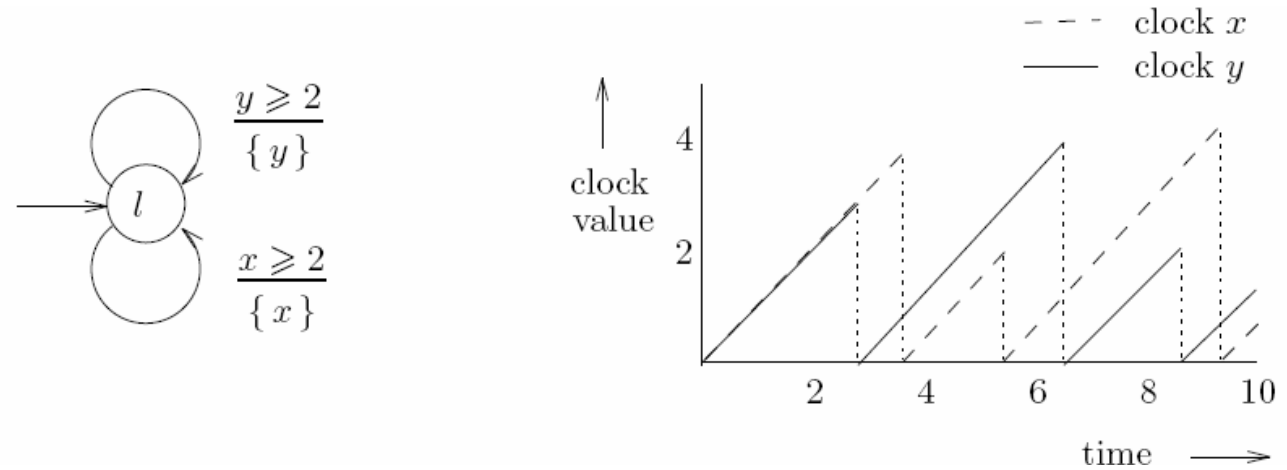


Рис. 4. Автомат с двумя таймерами

1.2. Алгоритм построения регионного автомата

Регионный автомат (модель Крипке) для автомата с таймерами строится на основе отношения эквивалентности между его состояниями.

Приведём пример построения модели Крипке для автомата с таймерами (рис. 5):

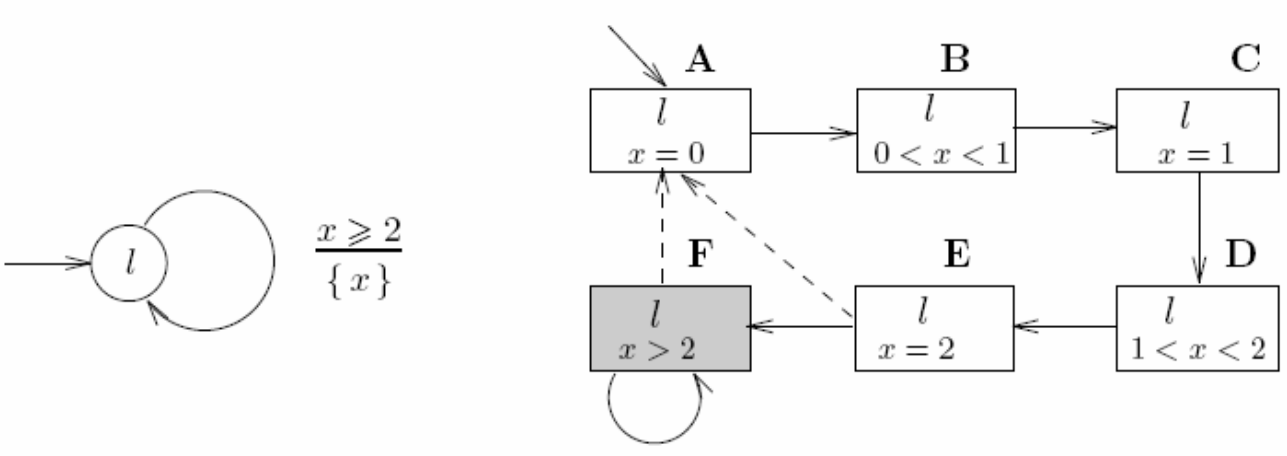


Рис. 5. Построение регионного автомата

В случае добавления одной формульной переменной автомат при его построении стандартным алгоритмом будет выглядеть следующим образом (рис. 6):

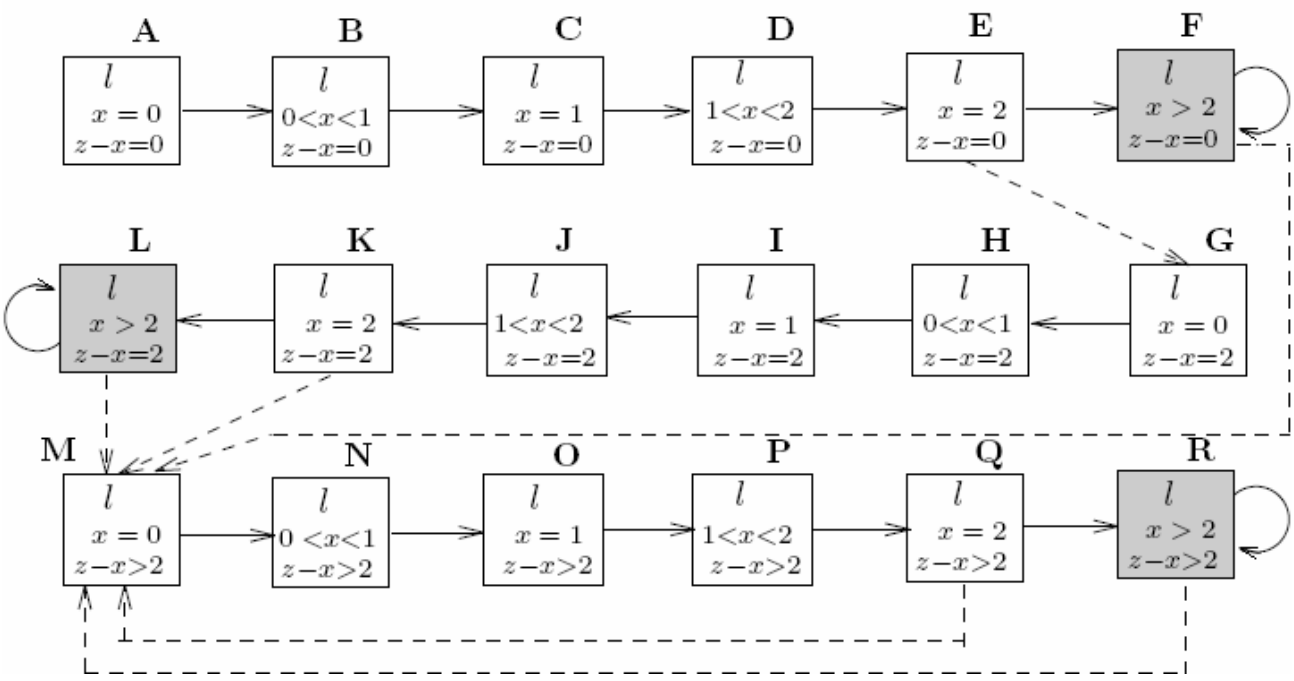


Рис. 6. Регионный автомат с одной формульной переменной

Как видно, автомат сильно разрастается. В частности, в нём содержится много линейных участков, следующих «змейкой друг за другом» (рис. 7):

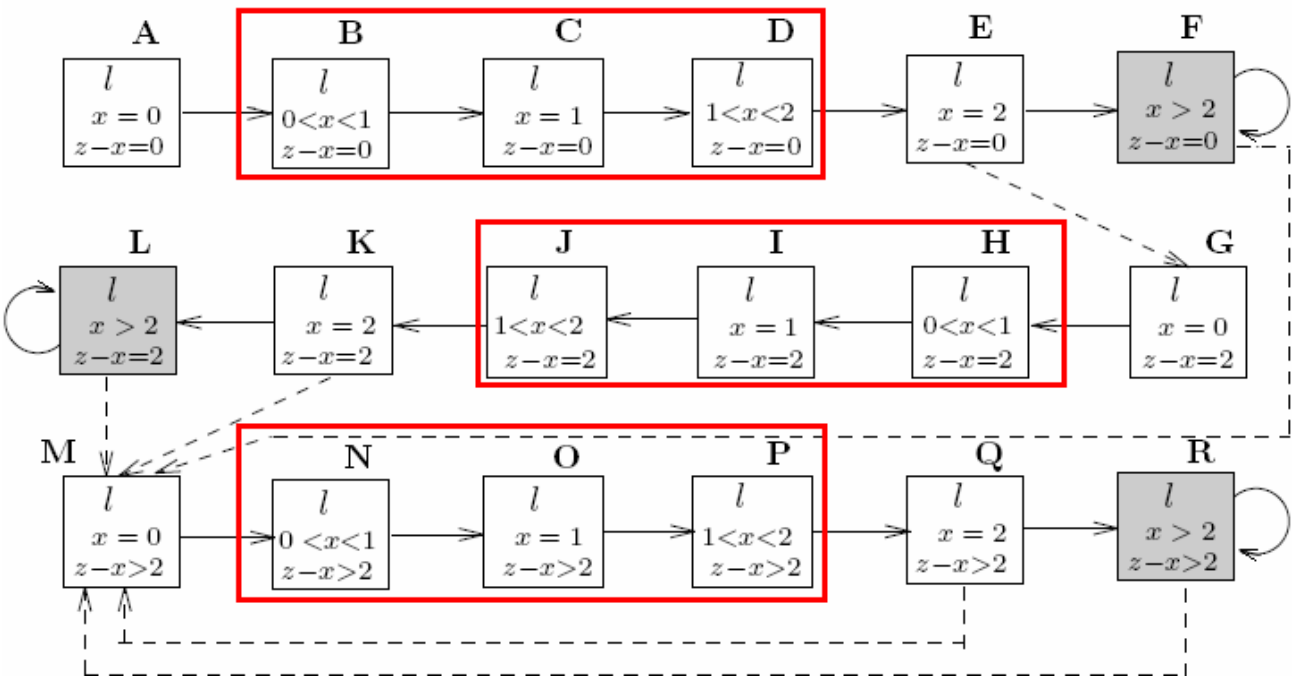


Рис. 7. Линейные цепочки состояний

Известный алгоритм [2] возвращает модель следующего размера: $O\left(n!\times 2^n \times \prod_{x\in\Psi}(c_x+1)\times|L|\right) = O\left(P(n)\times 2^n \times V \times |L|\right)$. Здесь $P(n)$ – число перестановок из n элементов, V – объём фазового пространства. При получении этой модели не учитывалась специфика множества классов эквивалентности, и её размер зависит от масштаба временных переменных.

Заметим, что все рёбра модели делятся на два класса: рёбра по переходу и рёбра по ожиданию. Все рёбра первого класса коллинеарны вектору $(1,\ldots,1)^T$, задающему направление течения времени. Множество классов эквивалентности разбивается условиями в формуле и автомате на (пересекающиеся) области.

Составим список всех множеств, полученных всеми возможными булевыми операциями над этими областями. Их будет в худшем случае $M = O(2^m)$, где m – число временных ограничений. Каждое из этих множеств спроецируем на плоскость $x_1 + x_2 + \ldots + x_n = 0$ (ортогональную оси времени). Плоскость изображена на рис. 8.

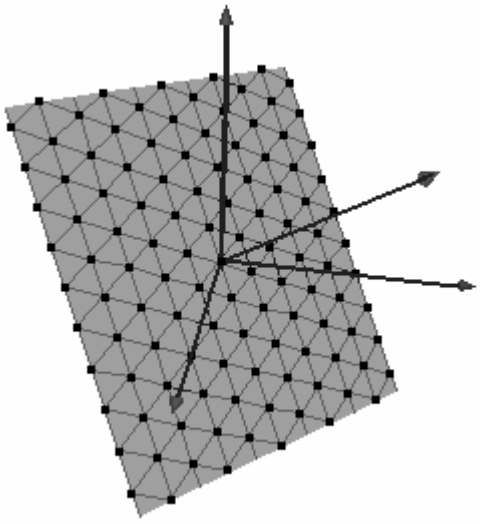


Рис. 8. Изометрическая плоскость

Для этого вычтем из каждой координаты проецируемой точки среднее арифметическое всех её координат. При этом получим набор (пересекающихся) образов множеств (рис. 9).

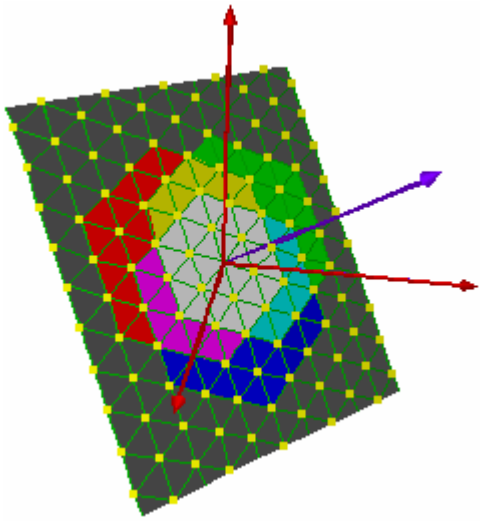


Рис. 9. Спроецированные образы множеств

Образы областей ограничений в изометрической проекции (рис. 10).

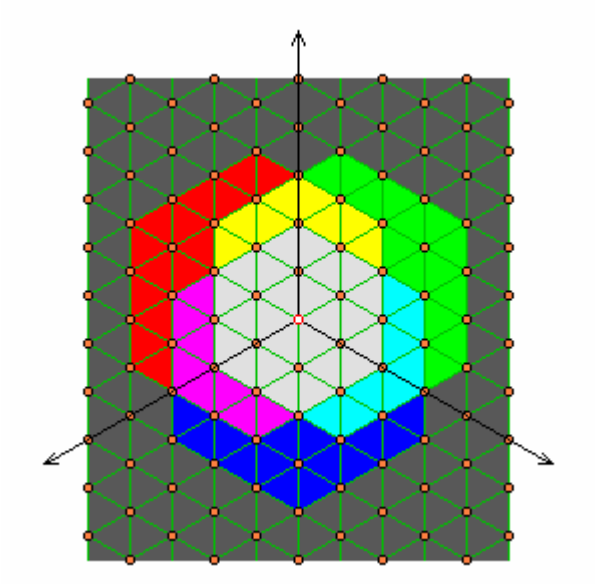


Рис. 10. Изометрическая проекция

Перекрывающиеся части образов соединяются рёбрами «по ожиданию» в порядке их удаления от начала координат. Число перекрывающихся частей ограничено оценкой $O(2^M)$.

Данный алгоритм строит модель с минимально возможным числом состояний (правда, линейные наборы из рёбер «по ожиданию» всё ещё возможны, но теперь они являются семантически значимыми, и их не обязательно удалять, хотя это и возможно при проведении дополнительного эвристического поиска). В случае малого числа рёбер «по ожиданию» соответствующий множитель в выражении для общего числа состояний можно будет оценить сверху не объёмом многомерного домена, а (n–1)-мерной площадью его проекции. Размер модели определяется соотношением:

$$O\left(P(n-1)\times 2^n \times S \times |L|\right) = O\left((n-1)!\times 2^n \times \prod_{x\in\Psi\setminus\{x_0\}}(c_x+1)\times|L|\right).$$

1.3. Верификация регионных автоматов

Рассматриваем в качестве примера автомат для выключателя (рис. 11).

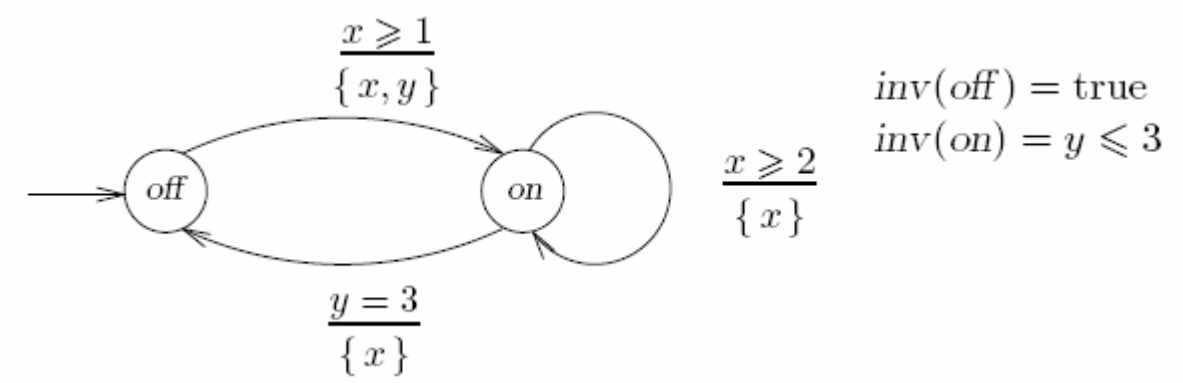


Рис. 11. Верифицируем автомат с таймерами

По нему строится регионный автомат (рис. 12).

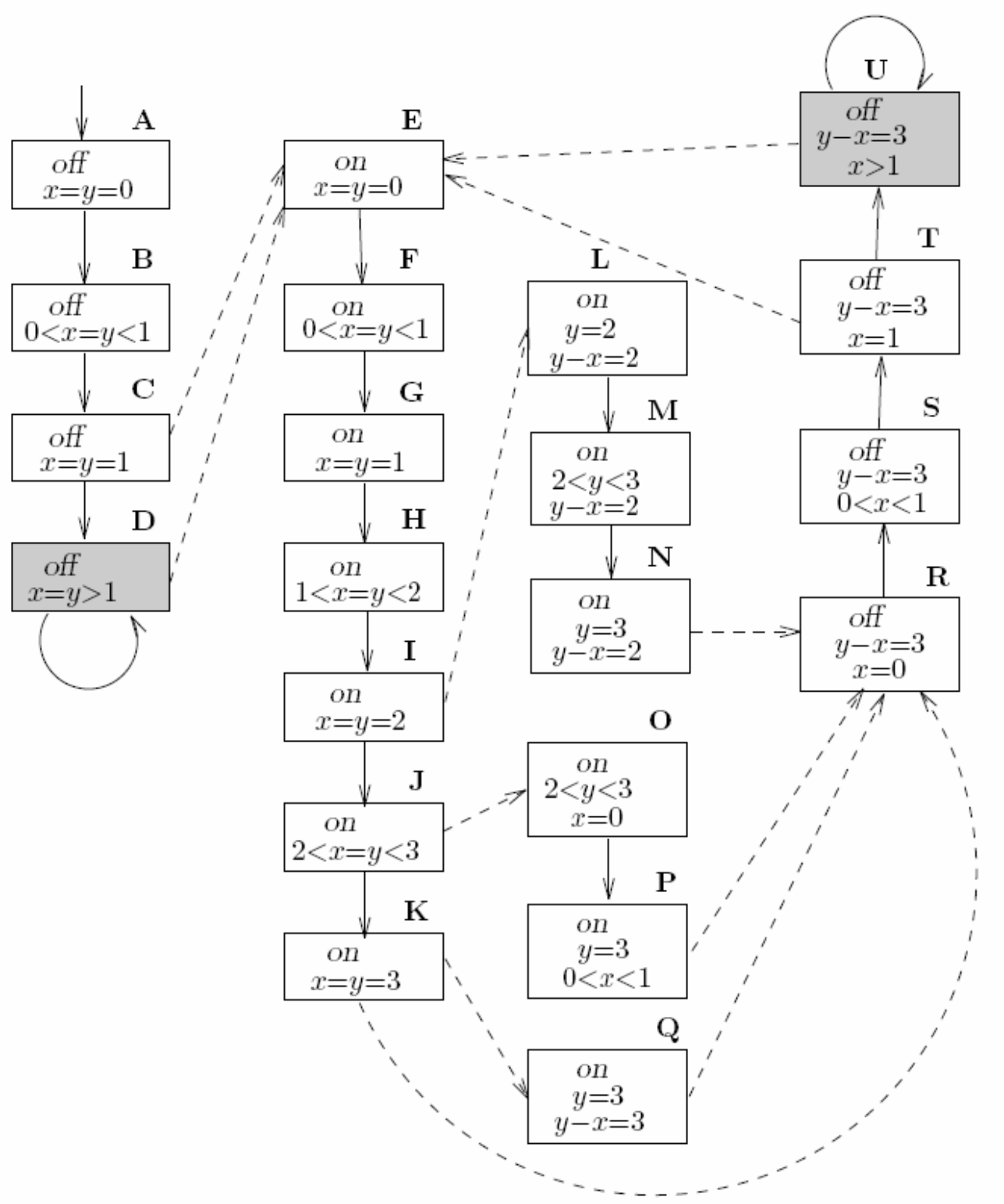


Рис. 12. Регионный автомат (модель Крипке),

полученный по исходному автомату с таймерами

Регионный автомат можно верифицировать по аналогии с *CTL*-моделью:

function $Sat^R(\phi : Formula) : \textbf{set of } Region;$
(* precondition: true *)
begin
$\quad$ **if** $\phi = \text{true} \longrightarrow$ **return** $S/_\approx$
$\quad$ [] $\phi = \text{false} \longrightarrow$ **return** $\varnothing$
$\quad$ [] $\phi \in AP \longrightarrow$ **return** $\{\, [s,w]_\approx \mid \phi \in Label(s) \,\}$
$\quad$ [] $\phi = \alpha \longrightarrow$ **return** $\{\, [s,w]_\approx \mid s = (l,v) \ \wedge\ v \cup w \models \alpha \,\}$
$\quad$ [] $\phi = \neg\, \phi_1 \longrightarrow$ **return** $S/_\approx - Sat^R(\phi_1)$
$\quad$ [] $\phi = \phi_1 \ \vee\ \phi_2 \longrightarrow$ **return** $(Sat^R(\phi_1) \cup Sat^R(\phi_2))$
$\quad$ [] $\phi = z.\phi_1 \longrightarrow$ **return** $\{\, [s,w]_\approx \mid (s,[z]w) \in Sat^R(\phi_1) \,\}$
$\quad$ [] $\phi = \mathsf{E}\,[\phi_1\, \mathsf{U}\, \phi_2] \longrightarrow$ **return** $Sat^R_{EU}(\phi_1,\phi_2)$
$\quad$ [] $\phi = \mathsf{A}\,[\phi_1\, \mathsf{U}\, \phi_2] \longrightarrow$ **return** $Sat^R_{AU}(\phi_1,\phi_2)$
$\quad$ **fi**
(* postcondition: $Sat^R(\phi) = \{\, [s,w]_\approx \mid \mathcal{M}, (s,w) \models \phi \,\}$ *)
end

function $Sat^R_{EU}(\phi, \psi : Formula) : \textbf{set of } Region;$
(* precondition: true *)
begin var $Q, Q' : \textbf{set of } Region;$
$\quad$ $Q, Q' := Sat^R(\psi), \varnothing;$
$\quad$ **do** $Q \neq Q' \longrightarrow$
$\quad\quad$ $Q' := Q;$
$\quad\quad$ $Q := Q \cup \big(\{\, s \mid \exists\, s' \in Q.\, s \longrightarrow s' \,\} \cap Sat^R(\phi)\big)$
$\quad$ **od**;
$\quad$ **return** Q
(* postcondition: $Sat_{EU}(\phi,\psi) = \{\, [s,w] \mid s, w \models \mathsf{E}\,[\phi\, \mathsf{U}\, \psi] \,\}$ *)
end

function $Sat^R_{AU}\,(\phi, \psi : \mathit{Formula})$: **set of** *Region*;
(* precondition: true *)
begin var Q, Q' : **set of** *Region*;
$Q, Q' := Sat^R(\psi), \varnothing$;
do $Q \neq Q' \longrightarrow$
$Q' := Q$;
$Q := Q \cup \big(\{\, s \mid \exists\, s' \in Q.\, s \longrightarrow s' \ \wedge\ (\forall\, s' \in Q.\, s \longrightarrow s')\,\} \cap Sat^R(\phi)\big)$
od;
return Q
(* postcondition: $Sat_{AU}(\phi, \psi) = \{\, [s, w] \mid s, w \models \mathsf{A}\,[\phi\,\mathsf{U}\,\psi]\,\}$ *)
end

Глава 2. Исследование задачи выполнимости

2.1. Введение в комбинаторные игры

Данная глава посвящена исследованию связи между задачей выполнимости и комбинаторными задачами на игровом поле с локальными взаимодействиями. Данный тип задач обладает следующими особенностями:

1) Существует специальный метод сведения задачи *SAT* к игровой задаче;

2) Существует метод решения игровой задачи более эффективный, чем полный перебор.

Примером такой задачи является игра «Сапёр» (*MINESWP*), которая будет рассмотрена в следующем разделе. Для решения игровых задач удобно применять динамическое программирование по профилю.

Опишем принцип сведения задачи выполнимости к игровой задаче:

$$SAT\ (q\text{-}SAT) \leq CIRCUIT\text{-}SAT\ (CIRCUIT - q\text{-}SAT) \leq GAME$$

Здесь сведение задачи SAT к схеме обозначено *CIRCUIT - SAT*.

Этот принцип сведения означает следующее: на основе булевой формулы строится схема из логических функциональных элементов, вычисляющая эту формулу. После этого каждый из функциональных элементов реализуется в качестве конфигурации игрового поля. Таким образом, построенная схема «изображается» на игровом поле в виде блоков, соответствующих функциональным элементам.

Пример булевой схемы, построенной по формуле

$$!(((p \,\&\, q) \,\&\, s \,\&\, !r) \,\&\, !(!s \,\&\, t)),$$

изображён на рис. 13.

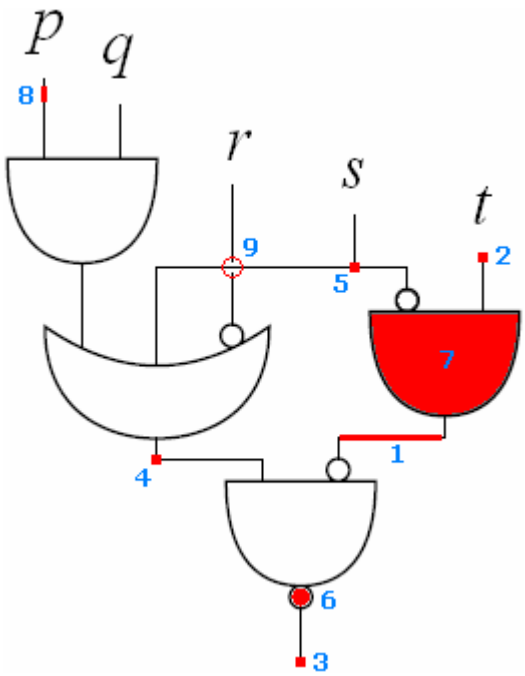


Рис. 13. Пример булевой схемы

Данная схема содержит девять функциональных элементов:

1) Проводник. Он предназначен для передачи (битового) сигнала между ячейками.

2) Вход. Это область, сигнал в которой может принимать значение 0 или 1.

3) Выход. В задаче выполнимости требуется узнать, существует ли такой набор входов, на котором выход будет равняться единице, следовательно, элемент «выход» – это такая область схемы, в которой сигнал определённо равняется единице.

4) Поворот. Это устройство служит для преобразования проводника, идущего в одном направлении (например, горизонтальному), в проводник, идущий в другом направлении (например, вертикальному). Здесь важно заметить, что схему совсем не обязательно укладывать в плоскости – для этого подходит пространство любой размерности. Поэтому в повороте можно использовать все степени свободы объемлющего пространства.

5) Ветвление. Этот элемент получает на вход некоторый проводник, а на выходе возвращает две копии этого проводника.

6) Инверсия. Обращает сигнал.

7) Конъюнкция – логическое «И».

8) Фазовый сдвиг. Участок, сдвигающий проводник на определённое число клеток игрового поля.

9) Пересечение (crossover). Пропускает проводники «друг сквозь друга» так, чтобы они друг другу не мешали. Применяется для случая непланарных схем.

В состав этого набора входят два логических элемента: шесть и семь. Эти типы элементов образуют базис для булевых функций – любую из них можно выразить только через эти две функции.

Элемент «Фазовый сдвиг» нужен для того, чтобы «подгонять» выходы одних функциональных элементов к входам других. Напомним, что игра рассматривается на дискретном клетчатом поле, и желательно иметь

устройство сдвига на число клеток, равное наибольшему общему делителю размеров остальных устройств (как правило, это единица).

Не всегда требуется находить конфигурации для *всех* рассмотренных элементов на языке игры. Некоторые из этих элементов можно выражать через другие. Например, выразим элемент 9 (пересечение) через элементы 1, 4 – 7 и, возможно, элемент 8. Пересечение можно рассматривать как чёрный ящик, действующий по схеме, изображённой на рис. 14.

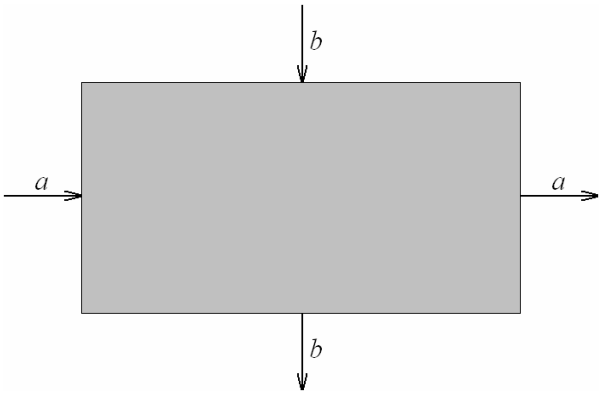


Рис. 14. Элемент «Пересечение»

Можно видеть, что пересечение представляет собой (выражается через) устройство, которое выдаёт оба входа, но только в обратном порядке (рис. 15).

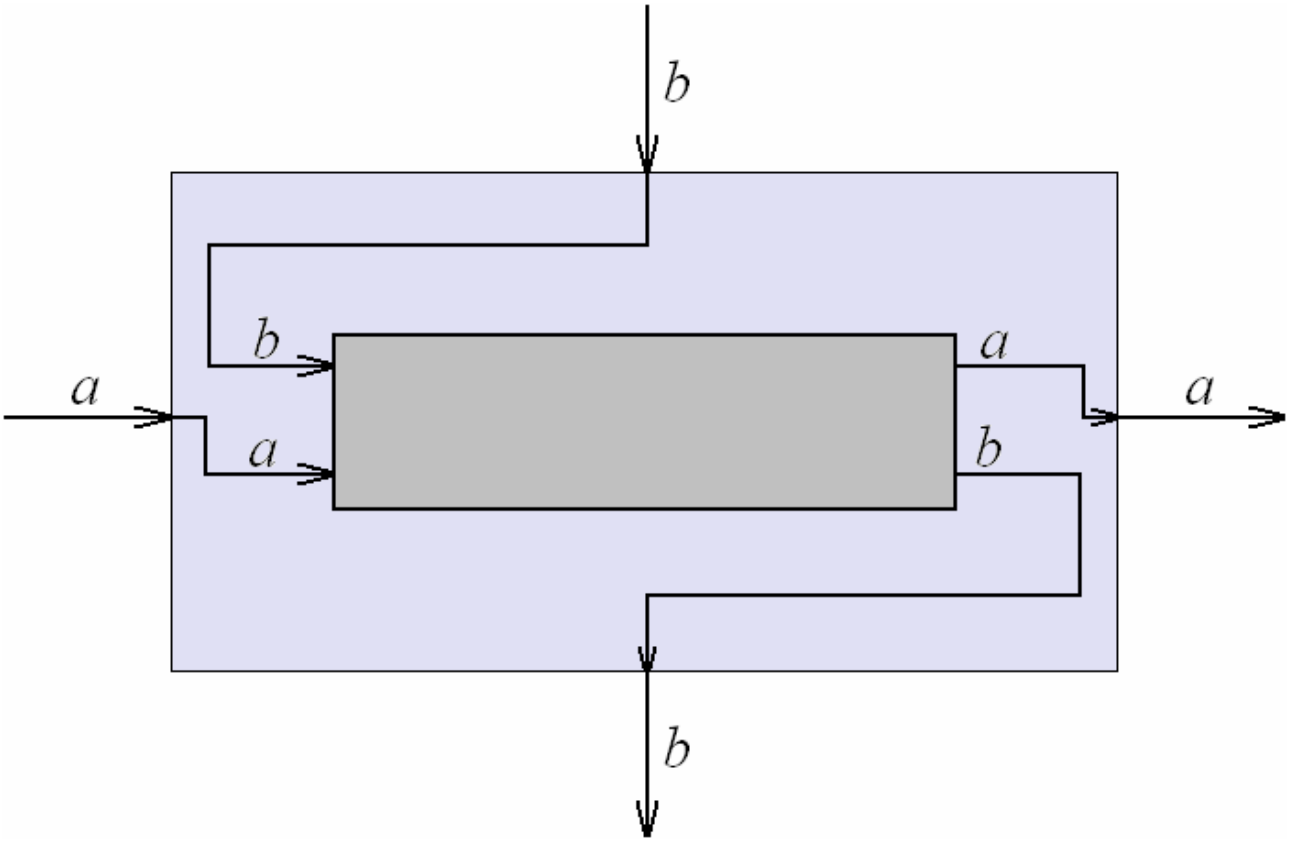


Рис. 15. Реализация пересечения

При этом внутреннее устройство (назовём его *swap*) должно быть реализовано без использования пересечения. Создание любого дополнительного (временного) проводника приводит к пересечению. Поэтому необходимо поменять значения *a* и *b* местами без использования дополнительного проводника. Это делается по аналогии со следующей последовательностью действий:

```
a := a ⊕ b;

b := a ⊕ b;

a := a ⊕ b;
```

Соответствующее устройство приведено на рис. 16.

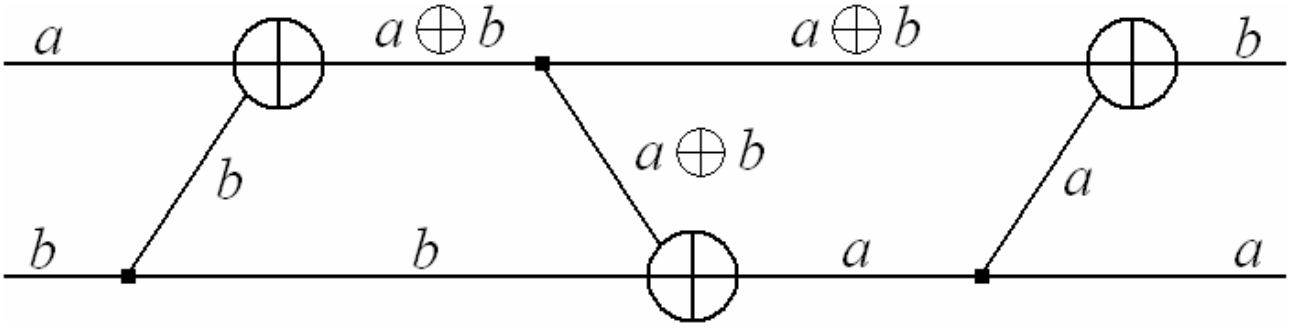


Рис. 16. Обмен сигналов местами

Осталось построить устройство (не использующее пересечения), реализующее операцию «сложение по модулю 2». Оно изображено на рис. 17.

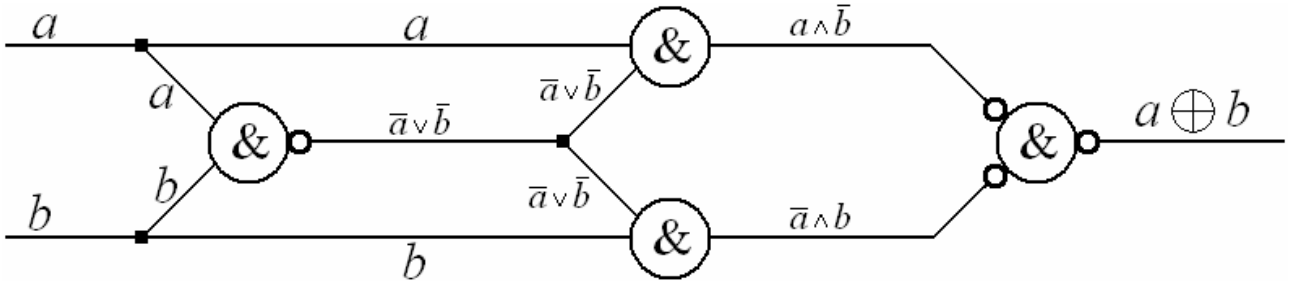


Рис. 17. Операция ⊕

Сведение задачи *SAT* к игре позволяет доказать *NP*-полноту этой игры (при условии, что корректность заполнения поля проверяется полиномиально – при условии, что игра принадлежит классу *NP*). Кроме того, это же сведение позволяет сделать вывод о Тьюринг-полноте игры, если расширить игровое поле до полубесконечного [27]. Под Тьюринг-полнотой понимается возможность эмулировать произвольную машину Тьюринга в терминах конфигураций игрового поля.

Действительно, рассмотрим некоторую машину Тьюринга. Без потери общности, что она имеет одну ленту, одну головку, бинарный алфавит и 2^k

состояний для некоторого натурального k. Тогда функция перехода этой машины – частично определённая функция, которая отображает пару (s, c) в тройку (s', c', *turn*). Здесь s и s' – начальное и конечное состояния, c и c' – начальный и конечный символы в ячейке ленты, *turn* – направление смещения головки. Поскольку домены всех переменных конечны, каждую переменную можно представить в виде двоичного набора, а саму функцию – в виде булевой. На основе такого представления можно определить «логический гейт» со входами и выходами, изображённый на рис. 18.

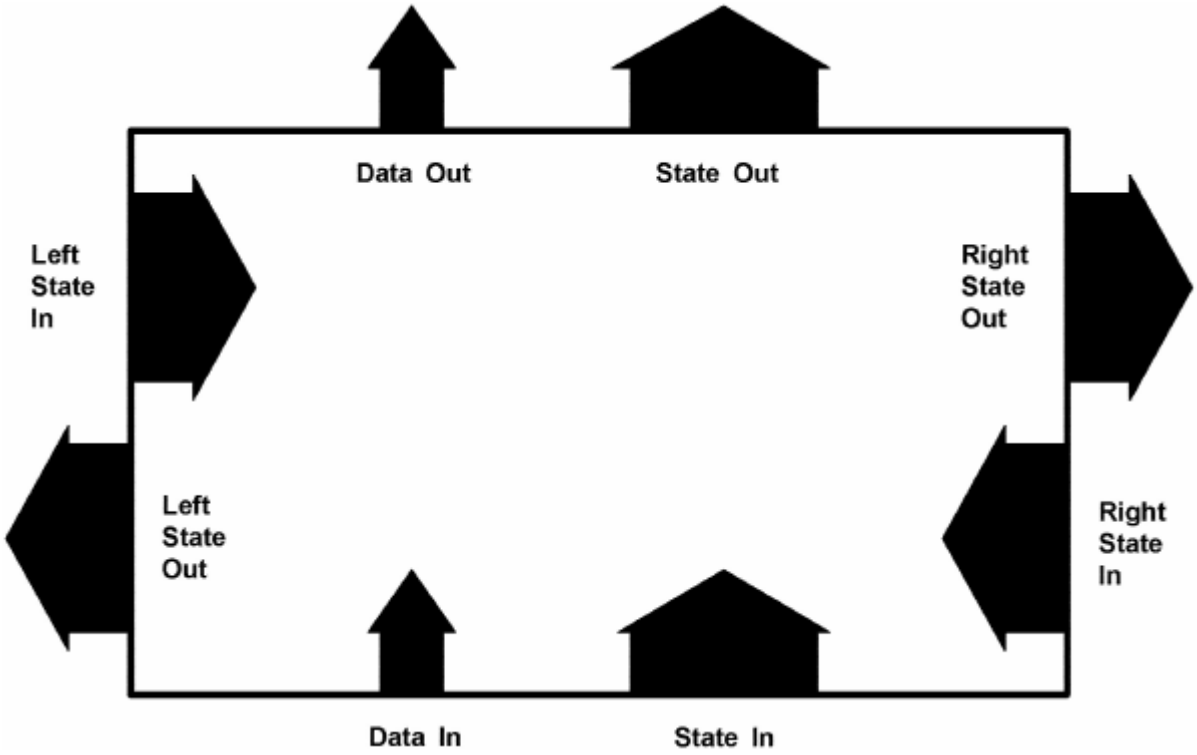


Рис. 18. Гейт для иллюстрации Тьюринг-полноты

Стрелки «Data In» и «Data Out» означают проводник, который несёт значение 0 или 1, записанное в позиции головки на ленте в текущий момент времени. Стрелки «State In» и «State Out» означают k проводников, которые

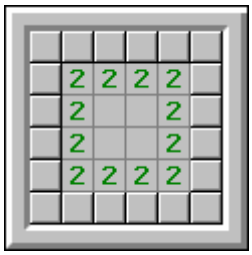
переносят двоичный номер состояния *s*. Если установить бесконечно много таких гейтов друг рядом с другом (заполнить ими полуплоскость), то каждый горизонтальный ряд в этом заполнении будет представлять собой *общее* состояние машины Тьюринга в фиксированный момент времени, и тем самым эмулировать её работу.

2.2. Задача «Сапёр»

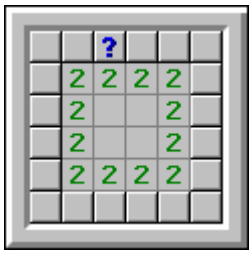
Условие задачи таково. Имеется прямоугольное поле, разбитое на клетки двух типов: открытые и закрытые. В каждой открытой клетке содержится целое неотрицательное число. Требуется определить, можно ли так заполнить некоторые закрытые клетки минами, чтобы число, стоящее в каждой открытой клетке, в точности равнялось числу «заминированных» соседей этой клетки.

Многие конфигурации игры «Сапёр», неразрешимые на первый взгляд без полного перебора, могут быть решены путём изощрённых логических рассуждений. Пример [20] приведён на рис. 19.

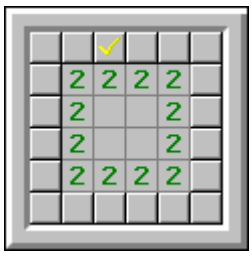
1. Рассмотрим следующее поле.



2. Содержит ли выделенная клетка мину?



3. Предположим, что она не содержит мину.



4. Тогда вследствие того, что обведённое значение равно двум, два его соседа заминированы.

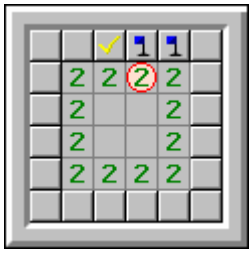
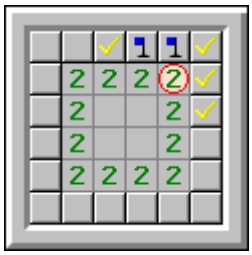
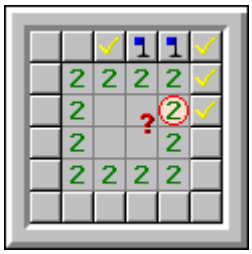


Рис. 19. Пример решения задачи

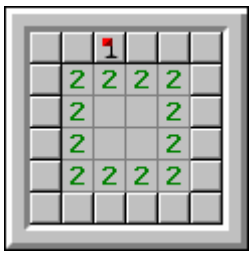
5. Следовательно, у сейчас обведённой клетки остальные соседи свободны.



6. Однако в этом случае число, записанное в клетке, обведённой теперь, приводит к противоречию.



7. Следовательно, выбранная вначале клетка содержит мину.



8. Поэтому симметричная ей клетка тоже содержит мину.

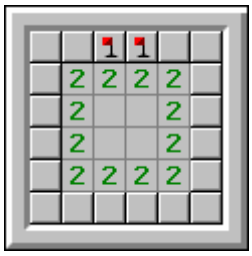
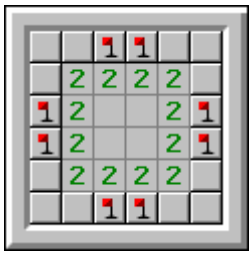


Рис. 19. Пример решения задачи (продолжение)

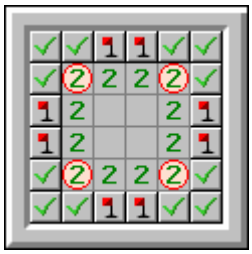
9. Всего у квадрата 8 симметрий ⇒

отмечено 8 клеток.



10. Обведённые клетки говорят о том,

что других мин нет.



11. Задача решена.

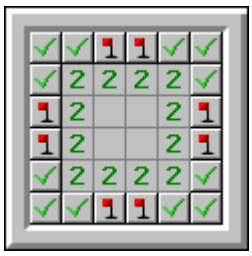


Рис. 19. Пример решения задачи (окончание)

Ещё одна сложность состоит в том, что в случае, когда требуется угадывать правильный ход, жадный алгоритм (выбирать клетку, вероятность появления мины в которой минимальна) не является оптимальным.

Рассмотрим такой пример (рис. 20):

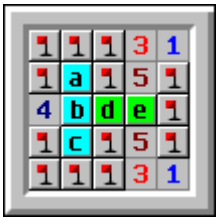


Рис. 20. Пример, когда жадный выбор не оптимален

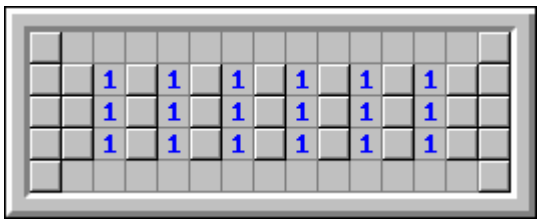
Тройка клеток $\{a, b, c\}$ содержит две мины, а двойка $\{d, e\}$ – одну мину. Следовательно, вероятность не попасть на мину, выбрав одну из клеток

$\{a, b, c\}$, равна 1/3, а вероятность не попасть на мину, выбрав одну из клеток $\{d, e\}$, равна 1/2. Поэтому можно сделать (неверный) вывод о том, что выбирать клетку *d* или *e* выгоднее. На самом деле выгоднее выбирать одну из клеток $\{a, b, c\}$, так как в этом случае, если (с вероятностью 1/3) не попасть на мину, то в выбранной клетке откроется число, которое позволит однозначно определить, содержит клетка *d* (а, следовательно, и *e*) мину или нет. Вероятность выигрыша в этом случае будет равна 1/3. Если же на первом ходе выбрать клетку *d* или *e*, то, даже если не попасть на мину, никакой информации это не даст, и вероятность выигрыша будет равна 1/2 * 1/3 = 1/6.

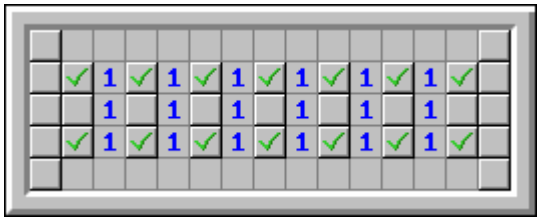
В данной работе вместо задачи *MINESWP* рассматривается более узкая задача: 1 - *MINESWP*. Задача *q* - *MINESWP* [32] отличается от задачи *MINESWP* тем, что в открытых клетках допускаются только числа, не превосходящие *q*. Очевидно, что *MINESWP* = *q* - *MINESWP* в случае, когда *q* равно числу соседей клетки на игровом поле (в плоском случае это восемь). Также очевидно, что задача 0 - *MINESWP* вырождена (так как поле без мин всегда удовлетворяет условию задачи). Далее будет выполнено сведение задачи *SAT* к задаче 1 - *MINESWP* (тем самым попутно будет показана *NP*-полнота этой задачи). Для этого построим девять функциональных элементов, позволяющих реализовать любую булеву схему на игровом поле.

Вначале построим проводник (рис. 21). Обратим внимание на его конструкцию: она такова, что вместе с проводником был построен логический элемент-инвертор (элемент 6).

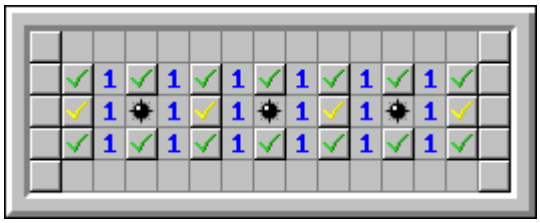
1. Рассмотрим следующую конфигурацию.



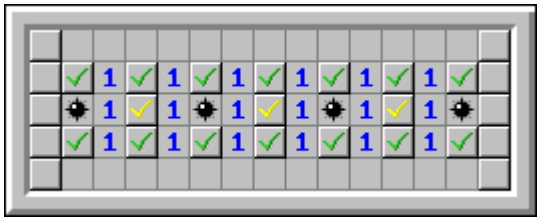
2. Крайние клетки пусты.



3. Одно из двух возможных решений.



4. Второе возможное решение.



5. Наличие двух возможных решений позволяет выделить проводник и записать передаваемый сигнал в виде булевой переменной.

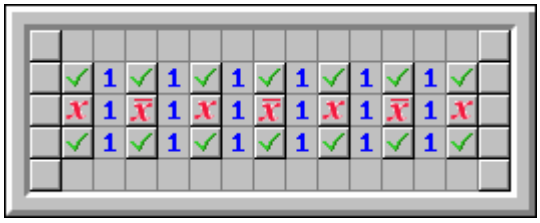


Рис. 21. Проводник (элемент 1) и инвертор (элемент 6)

На рис. 22 слева изображён вход (элемент 2), а справа – выход (элемент 3).

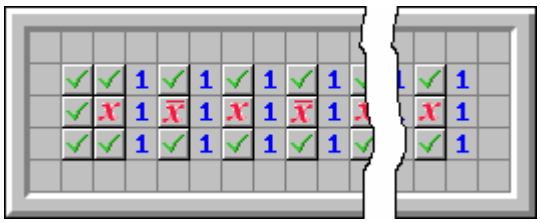


Рис. 22. Вход (элемент 2) и выход (элемент 3) логической схемы

Проводник имеет период 4. На рис. 23 слева изображено устройство, осуществляющее фазовый сдвиг (элемент 8), а справа можно увидеть периоды, на которые он разбивает проводник.

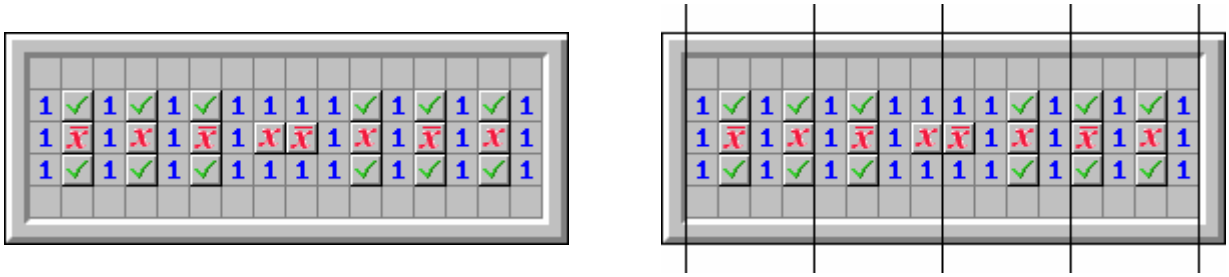


Рис. 23. Фазовый сдвиг (элемент 8)

На рис. 24 слева изображено устройство, осуществляющее поворот (элемент 4), а справа – устройство, разбивающее проводник на два (элемент 5).

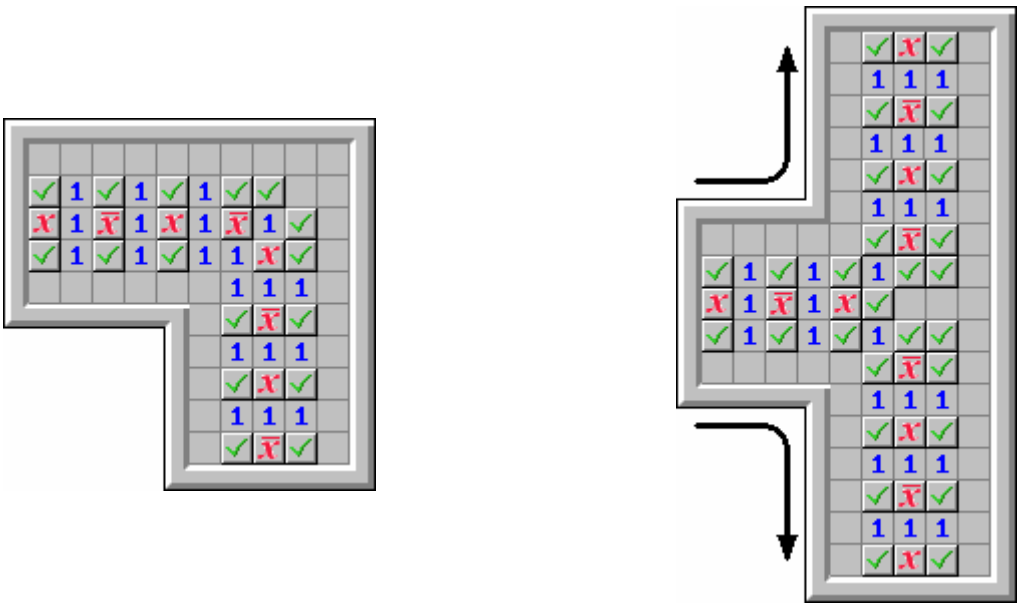


Рис. 24. Поворот (элемент 4) и ветвление (элемент 5)

На рис. 25 приведена реализация элемента под номером 7 – конъюнкции.

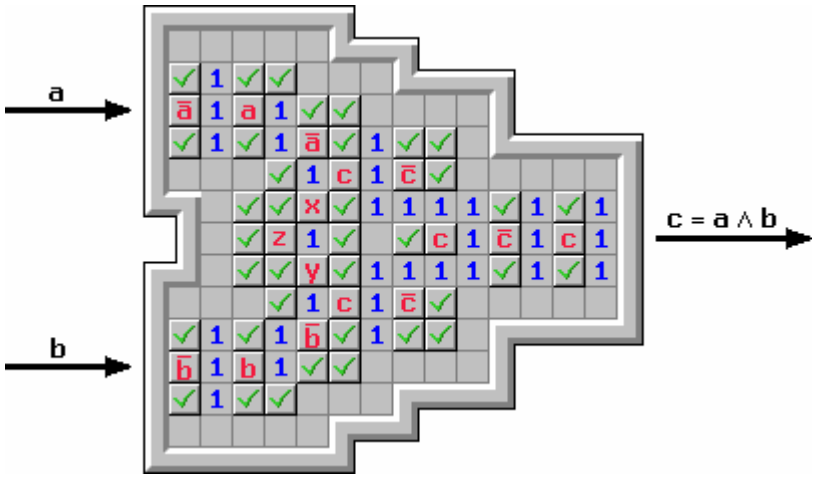


Рис. 25. Конъюнкция (элемент 7)

То, что данный элемент действительно реализован правильно, следует из того, что система булевых уравнений

$$\neg a + x + c = 1$$

$$\neg b + y + c = 1$$

$$x + y + z = 1$$

эквивалентна следующей системе:

$$x + c = a$$

$$y + c = b$$

$$a + b + z = 1 + 2c,$$

а у этой системы, в свою очередь, последнее уравнение приводит к формуле

$$c = a \wedge b.$$

Итак, сведение задачи *CIRCUIT - SAT* к задаче 1 - *MINESWP* выполнено.

Отметим также обобщение этой игры, имеющее название «Цветной Сапёр». Оно отличается от «Сапера» тем, что в её условии каждая закрытая клетка имеет некоторый цвет, и на конфигурации игрового поля накладывается следующее ограничение: все клетки одного цвета либо одновременно должны содержать мину, либо одновременно должны не содержать её. Эта игра намного сложнее, так как в ней нет локальных взаимодействий. Её *NP*-полнота может быть показана непосредственным сведением от задачи *1 – in – q - SAT* [33].

2.3. Динамика по профилю

Динамика по профилю представляет собой обход в ширину графа, соответствующего игровому полю. Клетки игрового поля являются его

вершинами, и любые две соседние клетки соединены ребром. На каждом шаге поддерживается и обновляется множество допустимых профилей.

Оценим временную сложность динамического программирования по профилю. Для этого введём следующие обозначения:

N(*k*) – (*Neighbours*) число новых (ещё не посещённых) соседей на *k*-й итерации динамического алгоритма.

NH(*k*) – (*Neighbours Hidden*) число новых *закрытых* соседей на *k*-й итерации динамического алгоритма.

NS(k) – (*Neighbours Single*) максимальное число новых соседей у *одной клетки* на *k*-й итерации динамического алгоритма.

Обозначения проиллюстрированы на рис. 26.

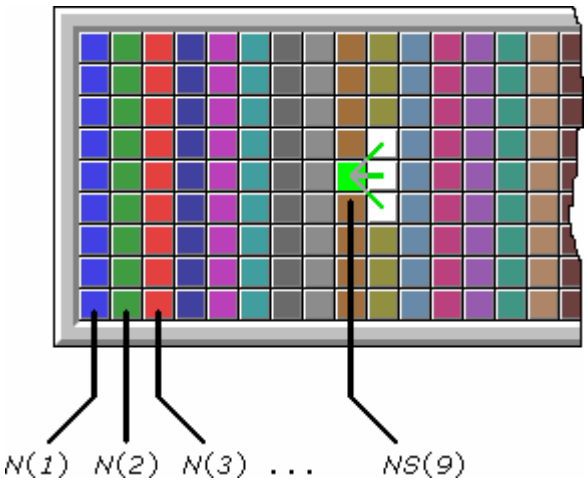


Рис. 26. Динамика по профилю

Пусть n – общее число итераций динамического алгоритма. Тогда время его работы $T(n)$ равно $\sum_{k=2}^{n-1} S(k)$, где $S(k)$ – время выполнения k-й итерации. При этом

$$S(k) \sim 2^{NH(k-1) + NH(k) + \min\{NH(k+1),\, NS(k)\cdot(N(k) - NH(k))\}} (N(k) - NH(k))\, NS(k).$$

Пусть $N(k) = O(f(k))$, и f не убывает.

Тогда $S(k) = O(2^{NH(k-1) + NH(k) + \min\{NH(k+1),\, NS(k)\cdot(N(k) - NH(k))\}} f^2(k)) =$

$$= O(2^{NH(k-1) + NH(k) + NH(k+1)} f^2(k)).$$

В общем случае $T(n) \le \sum_{k=2}^{n-1} 2^{f(k-1)+f(k)+f(k+1)} f^2(k)$.

Если $f(k) = O(1)$, то $T(n) = O(n)$ – в случае ограниченного числа новых соседей решение линейно. Примером является игровое поле размером $n \times m$, где n – переменная величина, а m – константа.

Усовершенствуем оценку. Если $N(k+1) - NH(k+1) > 2\log_2 f(k+1)$, то

$$S(k) = O(2^{NH(k-1) + N(k) - 2\log_2 f(k) + NH(k+1)} f^2(k)) =$$
$$= O(2^{NH(k-1) + N(k) + NH(k+1)}) = O(2^{f(k-1) + f(k) + f(k+1)}).$$

Если же $N(k+1) - NH(k+1) \le 2\log_2 f(k+1)$, то

$$S(k) = O\left(2^{NH(k-1)+NH(k)+NS(k)\cdot(N(k)-NH(k))} f^2(k)\right) =$$
$$= O\left(2^{f(k-1)+f(k)} 2^{2\log_2 f(k)\cdot NS(k)} f^2(k)\right) =$$
$$= O\left(2^{f(k-1)+f(k)} f^{2(NS(k)+1)}(k+1)\right).$$

Если при этом $NS(k) = O(1)$, то $S(k) = O(2^{f(k-1)+f(k)+f(k+1)})$.

Таким образом, из $NS(k) = O(1)$ в любом случае следует, что

$$T(n) \sum_{k=2}^{n-1} 2^{f(k-1)+f(k)+f(k+1)}.$$

Частным случаем этой формулы является грубая оценка:

$$T(n) = O(8^{f(n)} n).$$

Рассмотрим прямоугольное поле размера $n \times m$. Пусть $N = nm$ – число клеток в нём (длина входа к задаче). Интересен следующий вопрос: как n и m должны зависеть от N, для того чтобы динамика по профилю выполнялась за полиномиальное время? Зависимости должны удовлетворять условию $N = n(N)\, m(N)$. Все дальнейшие рассуждения выполняются для произвольного значения N или же для произвольного, начиная с некоторого. Справедливы соотношения:

$$T(N) = O\left(8^{m(N)} n(N)\right) = O\left(\frac{N}{m(N)} 8^{m(N)}\right)$$

$$T(N) = O\left(N^c\right) \Leftrightarrow m(N) = O(\log N) \Leftrightarrow \exists c > 0 \mid m(N) \le c \log N \Leftrightarrow$$

$$\Leftrightarrow \forall c' > 0 \ \exists c > 0 \mid m(N) \le c(\log N - \log(c' \log N)) = c \log \frac{N}{c' \log N} \Leftrightarrow$$

$$\Leftrightarrow m(N) \le c \log \frac{N}{m(N)} = c \log n(N) \Leftrightarrow m = O(\log n).$$

Следовательно, для полиномиальности необходимо и достаточно, чтобы поле имело размер порядка $n \times \log n$.

При этом если $m = \alpha \log_8 n$, то $T(N) = O(8^m n) = O(n^{1+\alpha})$.

2.4. Связь с задачей реализации графов

Выполним сведение задачи *q-SAT* к комбинаторной задаче на игровом поле 1-*MINESWP*. Пусть имеется конъюнкция, состоящая из *q*-клозов в числе *n* штук (общее число пропозициональных переменных обозначим через *m*). Тогда булева схема, полученная по этой формуле, будет иметь вид графа, в котором имеется два класса вершин. Первый из них состоит из *m* вершин, каждая из которых соответствует пропозициональной переменной, а второй – из *n* вершин, каждая из которых соответствует клозу (точнее, участку схемы, состоящему из $q - 1$ конъюнкции, которые реализуют данный клоз).

Из вершины первого класса будет вести ребро в вершину второго класса в том случае, если соответствующая переменная присутствует в соответствующем клозе (если она присутствует в нём с отрицанием, на ребре будет расположен инвертор). В результате получим ориентированный граф, в котором число рёбер, входящих в данную вершину ограничено числом *q*. Поскольку этот граф будет реализован на дискретном игровом поле, размер его функциональных элементов ограничен снизу, так как они не могут перекрываться. Таким образом, реализация рассматриваемого графа в *k*-мерном требует, чтобы размер вершин был не менее некоторой заранее определённой константы. То же самое следует сказать и о «толщине» рёбер – о размере их $(k-1)$-мерного поперечного сечения.

Пусть граф, соответствующий булевой схеме, уложен в область, имеющую характерный линейный размер (радиус, диаметр или длину стороны наименьшего содержащего её гиперкуба) l и объём $V \sim l^k$. В этом случае динамика по профилю для комбинаторной игры будет иметь следующую временную сложность: $T_k(n) = \tilde{O}\left(a^{l^{k-1}(n)}\right) = \tilde{O}\left(a^{V^{\frac{k-1}{k}}(n)}\right)$.

Существующие алгоритмы решения *SAT* имеют временную сложность $O(a^n)$ для некоторого a. Поэтому имеется смысл в поиске условий, при которых эту задачу можно решить более эффективно (на качественном уровне), например за время $\tilde{O}\left(a^{\sqrt{n}}\right)$, или, скажем, за $\tilde{O}\left(a^{n^{2/3}}\right)$, или, вообще, в широком смысле, за $\tilde{O}\left(a^{n^{\alpha}}\right)$, где $\alpha < 1$. При использовании метода, описанного в данной работе (сведение к задаче на игровом поле и решение этой задачи динамическим программированием по профилю), можно указать условия, при которых будет получена требуемая оценка сложности для решения *SAT*: эту задачу можно решить за $\tilde{O}\left(a^{n^{\alpha}}\right)$ ($\alpha < 1$), если выполняются вместе условия 1 и 2 или выполняется условие 3.

1. Комбинаторную игру типа *MINESWP* можно решить эффективнее, чем за $\tilde{O}\left(a^{l^{k-1}(n)}\right) = \tilde{O}\left(a^{V^{\frac{k-1}{k}}(n)}\right)$.

2. Граф, соответствующий булевой схеме, можно вложить в область, которая имеет *k*-мерный объём $V = O\left(n^{\frac{k}{k-1}} \right)$ с линейным размером $l = O\left(n^{\frac{1}{k-1}} \right)$.

3. Граф, соответствующий булевой схеме, может быть (более компактно) уложен в область, которая имеет *k*-мерный объём $V = o\left(n^{\frac{k}{k-1}} \right)$ с линейным размером $l = o\left(n^{\frac{1}{k-1}} \right)$.

Третья глава данной диссертации посвящена доказательству того, что условие 3 в большинстве случаев не выполняется (количество этих случаев также будет оценено). Это означает, что минимально возможный объём укладки имеет асимптотику $V = \Omega\left(n^{\frac{k}{k-1}} \right)$ (как выяснится, эта асимптотика подходит для худшего и для среднего случаев). Таким образом, упростив набор условий, можно утверждать, что задача *SAT* может быть решена более эффективно, если будет найден качественно более эффективный (например, по отношению к обычной динамике по профилю) алгоритм для решения комбинаторных игр и укладке графов в пространстве.

Глава 3. Оптимальные укладки графов

3.1. Общие сведения

В данной главе будут рассматриваться графы специального вида. Число вершин в них будем обозначать n. В каждую вершину будет входить ровно d рёбер. При реализации в пространстве вершины представляются в виде гипершаров фиксированного радиуса, а рёбра – в виде гиперкривых фиксированной толщины.

Будем говорить, что граф вложен в k-мерное пространство, если:

1) Каждой вершине графа сопоставлена точка этого пространства.
2) Каждой дуге сопоставлена непрерывная кривая, соединяющая вершины, которым она инцидентна.
3) Расстояние между точками, соответствующими любым двум различным вершинам, не меньше l_1.
4) Расстояние между кривыми, соответствующими любым двум неинцидентным рёбрам, не меньше l_2.

Предполагается, что значения l_1 и l_2 находятся в таком соотношении, которое позволяет рёбрам в числе d штук входить в одну и ту же вершину. Чаще всего удобно положить $d = 2$, $l_1 = l_2 = 1$. Будем считать, что эти параметры заданы таким образом.

Верхняя оценка на объём 3-мерного вложения графа имеет вид $V = O(n\sqrt{n})$ и основывается на работах [42, 43, 47–49].

Расположим все вершины в двумерном основании параллелепипеда, длина сторон которого равна $c\sqrt{n}$ для некоторой константы c, которую впоследствии необходимо будет подобрать. Вершины при этом будут образовывать сетку из прямоугольников. Константу c необходимо будет сделать настолько большой, чтобы расстояния между вершинами (период сетки) было достаточно велико для того, чтобы рёбра, входящие в одну вершину могли выполнять условия вложения – не пересекаться вне окрестности вершины радиуса l_1.

Далее будет приведён метод получения ортогональных вложений. Сопоставим каждому ребру (a, b) непрерывную ломаную линию $K_{a,b}(h)$, зависящую от дискретного параметра h, которая расположена в пространстве следующим образом: из вершины b исходит начальный сегмент этой ломаной. Этот сегмент находится целиком в двумерной грани рассматриваемого параллелепипеда. Начальные сегменты всех ломаных, входящих в одну и ту же вершину, исходят в разных направлениях таким образом, чтобы пересечь сферу радиуса l_1 с центром в рассматриваемой вершине в точках, находящихся друг от друга на расстояниях не меньше l_2. Следующий сегмент ломаной ортогонален гиперграни, в которой расположены вершины, и имеет длину h.

Можно сказать, что он «поднимается из этой грани на высоту h». Следующие сегменты этой ломаной расположены таким образом, чтобы на её конце были изменены все координаты по отношению к начальной вершине (изменяется по одной координате на сегмент). Новые значения координат должны быть такими, чтобы ни одна из новых координат не совпадала ни с одной старых координат вершин. После этого тем же самым методом пустим другую ломаную из вершины a, построив первые несколько её сегментов. В результате на «высоте» h будут находиться два конца построенных нами ломаных, которые осталось только соединить сегментами, каждый из которых изменяет очередную координату первого конца на соответствующую ему координату второго конца.

Пример получившегося ребра для трёхмерного случая изображён на рис. 27.

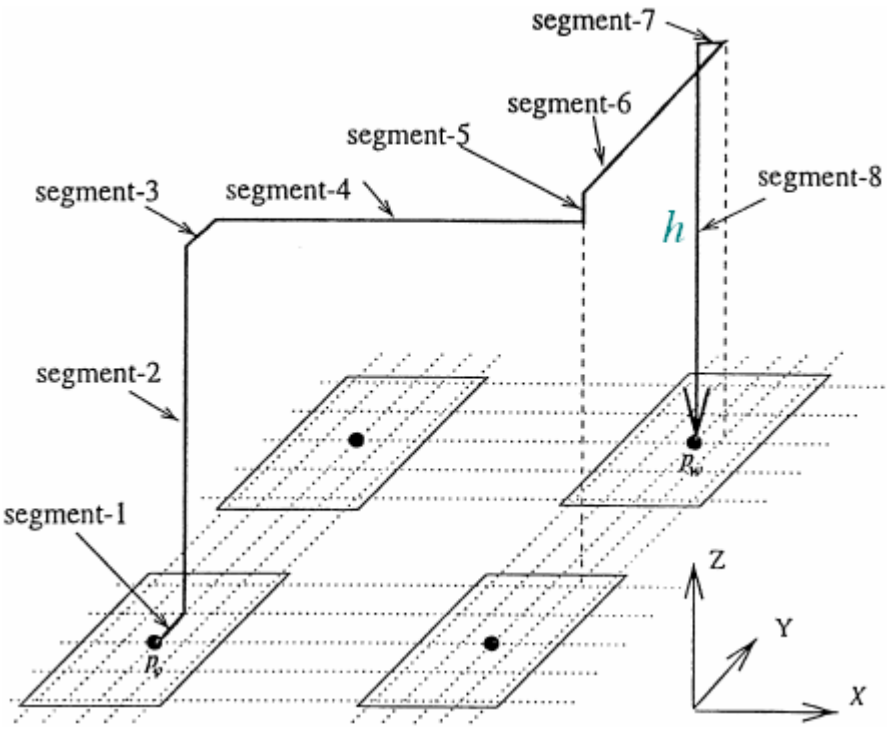


Рис. 27. Построение ребра на высоте *h*

Два ребра (a, b) и (a', b') назовём родственными, если точки a и a' имеют одинаковые абсциссы, или если точки b и b' имеют одинаковые ординаты. Если предположить, что число соседей вершины ограничено, то число рёбер, родственных заданному ребру, будет ограничено оценкой $O(\sqrt{n})$.

Способ построения ломаной для ребра имеет следующие свойства:

1) Если два ребра (a, b) и (a', b') не являются родственными, то соответствующие им ломаные $K_{a,b}(h)$ и $K_{a',b'}(h')$ при любых значениях параметров h и h' находятся друг от друга на расстоянии не меньше $l_2 = 1$ (за исключением окрестности вершины радиуса $l_1 = 1$, если эти рёбра инцидентны).

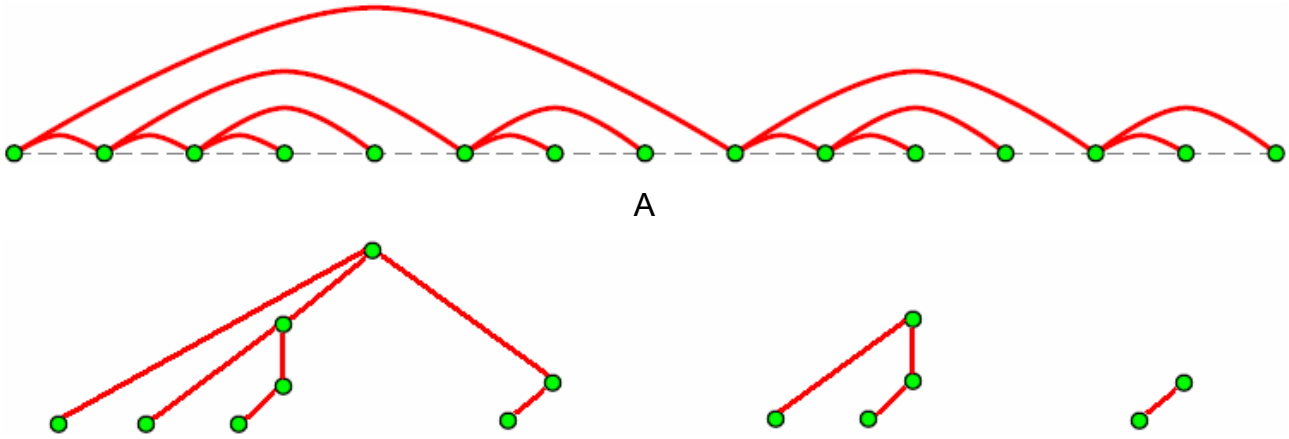
2) Если два различных ребра $(a,\ b)$ и (a', b') являются родственными и $h \neq h'$, то соответствующие им ломаные $K_{a,b}(h)$ и $K_{a',b'}(h')$ находятся друг от друга на расстоянии не меньше $l_2 = 1$ (за исключением окрестности вершины радиуса $l_1 = 1$, если эти рёбра инцидентны).

Поэтому осталось показать, что для рёбер графа можно подобрать дискретный набор значений параметра h так, чтобы эти значения для разных родственных рёбер были разными и все ломаные находились в параллелепипеде с длиной стороны не более $c\sqrt{n}$. Такой выбор значений параметра h возможен, потому что для любого ребра число рёбер, родственных ему, равно $O(\sqrt{n})$ –. не превышает числа допустимых различных значений параметра h.

Если же считать число рёбер, исходящих из вершины, неограниченным, то любой n-вершинный граф путём добавления линейного возрастающего числа новых вершин можно превратить в граф с ограниченной смежностью (при этом появятся ветвления). При вложении полученного графа в пространство точки ветвления необходимо интерпретировать как вершины.

Выше был рассмотрен трёхмерный случай. Оценки, полученные в данной работе, распространяются и на двумерный случай с оговоркой о том, что на граф накладывается условие планарности – он должен хоть как-то вкладываться в двумерное пространство.

В случае (плоских) *книжных вложений* граф всегда можно реализовать на площади, пропорциональной квадрату числа вершин. Каждому книжному вложению графа можно сопоставить «лес рёбер» – лес, вершинами которого являются рёбра нашего графа. В этом лесу из одной вершины ведёт ребро в другую в том и только в том случае, если ребро исходного графа, соответствующее первой вершине, в рассмотренном вложении объемлет ребро, соответствующее второй вершине, и никакое другое ребро не объемлет это последнее ребро. Более строго, можно ввести отношение «ребро *a* объемлет ребро *b* во вложении», являющееся отношением порядка. Лес рёбер – это диаграмма данного отношения порядка. Пример книжного вложения и леса рёбер, соответствующего ему, изображён на рис. 28.



А

Б

Рис. 28. Книжное вложение (а) и лес рёбер (б) для него

Длина ребра линейно зависит от максимального перепада его абсцисс и ординат. И то, и другое ограничено сверху линейной зависимостью от числа вершин, так как число рёбер зависит от числа вершин также линейно

(напомним про ограниченную смежность), то площадь, занимаемая графом, имеет оценку $O(n^2)$.

3.2. Оценка снизу

Не каждый граф требует большого объёма для своего вложения. Действительно, пусть $n = (d+1)\,m$, и граф состоит из m частей, не связанных друг с другом, каждая из которых представляет собой полный $(d+1)$-вершинный граф. Тогда $m \sim n$, и каждая из этих m частей занимает постоянный объём. Упаковывая эти части плотно друг рядом с другом, получим вложение в объём $O(n)$, линейный размер которого $O(\sqrt[k]{n})$.

Рассмотрим следующие интуитивные соображения. Если граф достаточно сложный, то у него много длинных рёбер (их длина имеет порядок характерного линейного размера области). Если выбрать некую поверхность, то эти рёбра будут пересекать эту поверхность в дизъюнктных эллипсоидах размерности $k-1$, постоянного объёма. Отсюда следует, что площадь поверхности должна быть примерно пропорциональна числу рёбер (а, следовательно, и вершин). Действительно, на интуитивном уровне оценку можно получить достаточно простыми соображениями:

$$l^k \approx nl \Rightarrow n \approx l^{k-1} \Rightarrow l \approx n^{\frac{1}{k-1}} \Rightarrow V \approx n^{\frac{k}{k-1}}.$$

Пусть w – некоторое разбиение вершин графа на три упорядоченные части, которые обозначим, соответственно, через w_1, w_2, w_3. Разбиение w графа

с n вершинами назовём (ε, δ)-разбиением, если w_1 содержит $[\varepsilon n]$ вершин, w_2 содержит $[\delta n]$ вершин и w_3 содержит $n - [\varepsilon n] - [\delta n]$ вершин. Под степенью связности $S(w)$ графа относительно разбиения будем понимать максимальное число неинцидентных рёбер, которые идут из вершин w_3 к w_1.

Теорема 1. Для любых положительных a_0, b (для которых $a_0 + b < 1$) существует такое ε_0, что для любых $\varepsilon \le \varepsilon_0$ и $a \le a_0$ степень связности почти всех n-вершинных графов относительно любого $(\varepsilon, a\varepsilon)$-разбиения не меньше $b\varepsilon n$.

Теорема 1 будет доказана в следующем разделе. Константы a_0 и b в ней можно фиксировать. Графы, удовлетворяющие её условиям, будем называть *универсальными*. Она позволяет сделать следующие два наблюдения:

1) Расстояние между любыми двумя неинцидентными рёбрами не меньше единицы.

2) У почти всех n-вершинных графов любые $\varepsilon_0 n$ вершин связаны с остальными вершинами с помощью как минимум $b\varepsilon_0 n$ неинцидентных дуг.

Отсюда вытекает следующее утверждение.

Утверждение. Почти все n-вершинные графы обладают следующим свойством: при любой реализации такого графа в k-мерном пространстве в любом гиперкубе с $(k-1)$-мерной площадью поверхности $S \le cb\varepsilon_0 n$ (для некоторого c, которое будет подобрано позже) находится меньше, чем $\varepsilon_0 n$ вершин.

Теорема 2. Для сетей, удовлетворяющих условиям Теоремы 1 и Утверждения минимальный объём укладки имеет асимптотику $V = \Omega\left(n^{\frac{k}{k-1}} \right)$.

Доказательство. Пусть μ и s – некоторые числа (их значения подберём позже). Рассмотрим некоторый граф с n вершинами, удовлетворяющий условию теорем 1 и 2 (такие графы – почти все). Пусть граф реализован в k-мерном пространстве и пусть T – некоторый содержащий его параллелепипед с длиной рёбер не менее $s\mu n^{\frac{1}{k-1}}$.

Тогда для каждого j от 1 до k с помощью гиперплоскостей $x_j = i\mu n^{\frac{1}{k-1}}$ (i = 0, 1, 2…) разбиваем параллелепипед T на параллельные слои толщины $\mu n^{\frac{1}{k-1}}$ (без ограничения общности будем считать, что последний слой тоже имеет толщину $\mu n^{\frac{1}{k-1}}$). Перенумеруем эти слои в направлении оси x_j числами 1, 2, 3, … Затем на основании этого разбиения строим разбиение параллелепипеда T на части $A_1^j, A_2^j, \ldots, A_s^j$. К A_i^j относим те и только те слои, которые имеют номер вида $ks + i$, где k – целое число. Затем из совокупности этих частей выбираем ту, которая содержит наименьшее число вершин графа. Пусть это будет часть A^j. Видно, что A^j содержит не больше чем n/s вершин. Слои, принадлежащие A^j, будем называть

отмеченными. Отсюда следует, что отмеченные слои параллелепипеда *T,* содержат не больше, чем kn/s вершин.

Пусть теперь числа μ и s такие, что гиперкубики, ограниченные отмеченными слоями, имеют $(k-1)$-мерную площадь поверхности не больше $cb\varepsilon_0 n$ (имеется в виду каждый кубик в отдельности) – пусть μ и s удовлетворяют неравенству:

$$2k\,(s-1)^{k-1}\,\mu^{k-1} n \le cb\varepsilon_0 n. \qquad (1)$$

Тогда согласно Утверждению каждый из упомянутых гиперкубиков содержит меньше, чем $\varepsilon_0 n$ вершин. Отсюда, в свою очередь, следует, что если число вершин, содержащихся в неотмеченных слоях, не меньше $\varepsilon_0 n$:

$$n - kn/s \ge \varepsilon_0 n, \qquad (2)$$

то можно построить такое множество этих гиперкубиков G, которое содержит не меньше чем ½ $\varepsilon_0 n$ и не больше чем $\varepsilon_0 n$ вершин.

Иллюстрация того, как область, в которую уложен граф, разбивается на слои приведена на рис. 29.

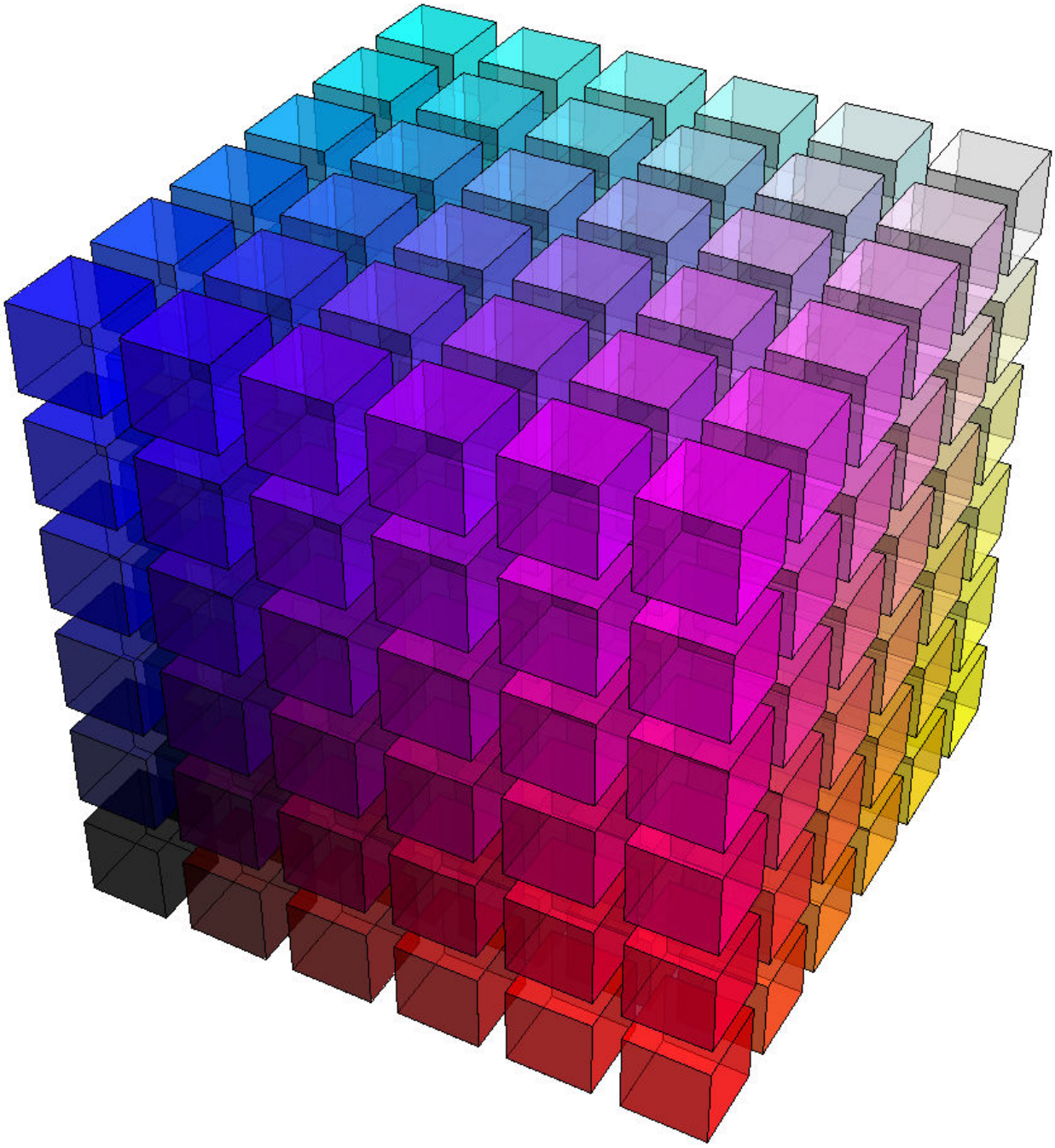


Рис. 29. Разбиение на слои

Пусть, далее, число s такое, что отмеченные слои содержат не более чем $a_0\ (\varepsilon_0/2)\ n$ вершин. Следовательно, пусть s удовлетворяет неравенству:

$$kn/s \leq a_0\ (\varepsilon_0/2)\ n. \qquad (3)$$

Рассмотрим следующее (ε, δ)-разбиение графа:

1) w_1 – это множество тех вершин графа, которым при рассматриваемой его реализации сопоставлены точки из G. Число элементов εn множества w_1 равно числу вершин, содержащихся в G. При этом $\varepsilon_0/2 \le \varepsilon \le \varepsilon_0$.

2) w_2 – это множество тех вершин графа, которым при рассматриваемой его реализации сопоставлены («мусорные») точки из отмеченных слоёв. Таким образом, число элементов δn множества w_2 равно числу вершин, содержащихся в отмеченных слоях. При этом $\delta \le a_0\varepsilon_0/2$.

По теореме 1 получим, что граф при таком (ε, δ)-разбиении имеет степень связности $\ge b\varepsilon n \ge b(\varepsilon/2)n$. Это означает, что вершины, содержащиеся в множестве кубиков G, и вершины, содержащиеся в остальных кубиках, ограниченных отмеченными слоями, соединены между собой по меньшей мере $b(\varepsilon_0/2)n$ неинцидентными рёбрами. Множество этих рёбер обозначим через E. Согласно определению вложения расстояние между рёбрами должно быть не меньше единицы. Так как толщина отмеченных слоёв равна $\mu n^{\frac{1}{k-1}}$, то каждое ребро множества E соединяет вершины, находящиеся на расстоянии не менее $\mu n^{\frac{1}{k-1}}$. Отсюда получаем, что любое тело, содержащее все рёбра из E (и при том так, что они находятся от его поверхности на расстоянии не меньше единицы), имеет объём не меньше, чем $b_0(\varepsilon/2)n\mu n^{\frac{1}{k-1}}Surf$ (здесь $Surf$ –

объём (k–1)-мерной гиперсферы радиуса 1/2). Теперь вспомним, что эти оценки были получены при условии, что s и μ удовлетворяют неравенствам (1) – (3). Можно убедиться, что существуют s и $\mu > 0$, не зависящие от n, которые удовлетворяют этим неравенствам.

Теорема 2 доказана.

3.3. Оценка числа графов

В этом разделе приводится доказательство Теоремы 1.

Теорема 1. Для любых положительных a_0, b (для которых $a_0 + b < 1$) существует такое ε_0, что для любых $\varepsilon \le \varepsilon_0$ и $a \le a_0$ степень связности почти всех n-вершинных графов относительно любого $(\varepsilon, a\varepsilon)$-разбиения не меньше $b\varepsilon n$.

Под степенью инцидентности $Z(w)$ относительно разбиения w будем понимать число вершин, принадлежащих w_3, из которых идут дуги к вершинам, принадлежащим w_1. Отсюда следует, что $S(w) \ge Z(w) / d$. Таким образом, теорема будет доказана, если будет доказано аналогичное утверждение для степеней инцидентности.

С этой целью процесс построения n-вершинных графов представим следующим образом. Вначале дано n вершин таких, что в каждую из них входит ровно d дуг со свободными вторыми концами. Затем свободный конец каждой дуги случайно подсоединяем к какой-нибудь вершине. В результате получаем некоторый граф. Пусть вначале было фиксировано некоторое (ε, δ)-разбиение w множества вершин. Обозначим через $P_w(\varepsilon, \delta, c, n)$ вероятность

того, что в результате описанной процедуры получим граф, который имеет степень инцидентности $Z(w) < cn$. Заметим, что при фиксированных ε и δ вероятность $P_w(\varepsilon, \delta, c, n)$ для всех (ε, δ)-разбиений множества вершин одна и та же и что число всевозможных (ε, δ)-разбиений множества вершин равно $C_n^{\varepsilon n} C_{n-\varepsilon n}^{\delta n}$. Обозначим через $P(\varepsilon, \delta, c, n)$ вероятность того, что в результате описанной выше процедуры получим граф, который хотя бы при одном (ε, δ)-разбиении имеет степень инцидентности $< cn$. Из сказанного выше следует, что $P(\varepsilon, \delta, c, n) < C_n^{\varepsilon n} C_{n-\varepsilon n}^{\delta n} \; P_w(\varepsilon, \delta, c, n)$.

Утверждение теоремы будет доказано, если докажем следующее утверждение: для любых положительных a_0, b (для которых $a_0 + b < 1$) существует такое ε_0, что для любых $\varepsilon \le \varepsilon_0$ и $a \le a_0$

$$P(\varepsilon, a\varepsilon, b\varepsilon, n) \to 0 \text{ при } n \to \infty.$$

Оценим $P_w(\varepsilon, \delta, c, n)$. С этой целью рассмотрим следующую вероятностную модель. Имеется n ящиков, среди которых εn белых, δn чёрных и $(1 - \varepsilon - \delta)n$ красных. Рассмотрим серию испытаний, состоящую в последовательном случайном бросании $d\varepsilon n$ шариков в ящики (одно испытание – это бросание одного шарика). Под благоприятным исходом испытания будем понимать попадание шарика в пустой красный ящик. Вероятность $P_w(\varepsilon, \delta, c, n)$ будет равна вероятности того, что число благоприятных исходов в серии испытаний будет $< cn$ (в роли шариков выступают свободные концы дуг, входящих в вершины w_1, в роли белых

ящиков – вершины w_1, в роли чёрных ящиков – вершины w_2, в роли красных ящиков – вершины w_3).

Введем следующие обозначения.

$$t \mapsto d\varepsilon n, \quad u \mapsto (1-\varepsilon-\delta)n, \quad s \mapsto cn$$

Тогда эта величина подчиняется рекуррентному закону:

$$P_n(t,u,s) = \frac{u}{n} P_n(t-1,u-1,s-1) + \left(1-\frac{u}{n}\right) P_n(t-1,u,s)$$
$$P_n(0,u,s) = P_n(t,0,s) = P(t,u,s)_{s\geq t} = P(t,u,s)_{s\geq u} = 1$$
$$P_n(t,u,0) = \left(1-\frac{u}{n}\right)^t$$
$$s < t < u$$

Сделаем подходящую замену переменной.

$$fix: u_0 = u - s$$
$$a_{i,j,u_0} = \frac{u_0+i}{n} a_{i-1,j-1,u_0} + \left(1-\frac{u_0+i}{n}\right) a_{i,j-1,u_0}$$
$$a_{i,0,u_0} = 1, \quad a_{0,j,u_0} = \left(1-\frac{u_0}{n}\right)^j$$

Полученное уравнение можно переписать уравнение в операторном виде:

$$\Delta_j a_{i,j} + \frac{u_0+i}{n}\left(a_{i,j} - a_{i-1,j}\right) = 0$$

Это уравнение решим методом Буля [85–87]:

$$a_{i,j} = \frac{(u_0+i)!(n\Delta_j+u_0+1)!}{(u_0+1)!(n\Delta_j+u_0+i)!}\varphi(j) = A^i_{u_0+i}\left(A^i_{n\Delta_j+u_0+i}\right)^{-1} a_{0,j} = A^i_{u_0+i}\left(A^i_{n\Delta_j+u_0+i}\right)^{-1}\left(1-\frac{u_0}{n}\right)^j$$

Запишем последовательность шагов решения:

$$k=\overline{1;i}: \quad \left(n\Delta_j+u_0+k\right)^{-1}$$
$$g(j)=(u_0+k)f(j)+n\left(f(j+1)-f(j)\right)$$

Сделаем первые четыре итерации:

$$\left(1-\frac{u_0}{n}\right)^j+c_1\left(1-\frac{u_0+1}{n}\right)^j$$
$$\frac{1}{2}\left(1-\frac{u_0}{n}\right)^j+c_1\left(1-\frac{u_0+1}{n}\right)^j+c_2\left(1-\frac{u_0+2}{n}\right)^j$$
$$\frac{1}{6}\left(1-\frac{u_0}{n}\right)^j+\frac{c_1}{2}\left(1-\frac{u_0+1}{n}\right)^j+c_2\left(1-\frac{u_0+2}{n}\right)^j+c_3\left(1-\frac{u_0+3}{n}\right)^j$$
$$\frac{1}{24}\left(1-\frac{u_0}{n}\right)^j+\frac{c_1}{6}\left(1-\frac{u_0+1}{n}\right)^j+\frac{c_2}{2}\left(1-\frac{u_0+2}{n}\right)^j+c_3\left(1-\frac{u_0+3}{n}\right)^j+c_4\left(1-\frac{u_0+4}{n}\right)^j$$

Удобно на этом шаге присвоить $c_0 := 1$.

В итерациях легко прослеживается закономерность. Обнаружив её, запишем сумму, которая тривиально доказывается по индукции:

$$a_{i,j}=A^i_{u_0+i}\sum_{k=0}^{i}\frac{c_k}{(i-k)!}\left(1-\frac{u_0+k}{n}\right)^j$$

Теперь решим уравнения, соответствующие начальным условиям:

$$a_{i,0}=A^i_{u_0+i}\sum_{k=0}^{i}\frac{c_k}{(i-k)!}=1$$

Сделаем первые пять итераций:

$$c_0=1,\quad c_1=-\frac{u_0}{u_0+1},\quad c_2=\frac{u_0}{2(u_0+2)},\quad c_3=-\frac{u_0}{6(u_0+3)},\quad c_4=\frac{u_0}{24(u_0+4)},\quad c_5=-\frac{u_0}{120(u_0+5)}$$

Аналогично, прослеживается закономерность, доказываемая по индукции:

$$c_k = \frac{(-1)^k u_0}{k!(u_0+k)}$$
$$u_0 \neq 0 \Rightarrow c_0 = 1$$

Теперь выпишем целиком новое уравнение:

$$a_{i,j,u_0} = C^i_{u_0+i} \sum_{k=0}^{i} (-1)^k C^k_i \frac{\left(1-\frac{u_0+k}{n}\right)^j u_0}{u_0+k}$$
$$u_0 \sum_{k=0}^{i} \frac{(-1)^k C^k_i}{u_0+k} = \left(C^i_{u_0+i}\right)^{-1}$$

Сделаем необходимые подстановки для введённых в начале обозначений:

$$P_n(t,u,s) = a_{s,t,u-s} = C^s_u \sum_{k=0}^{s} (-1)^k C^k_s \frac{\left(1-\frac{u-s+k}{n}\right)^t}{1+\frac{k}{u-s}}$$
$$P(d,n,\varepsilon,\delta,c) = P_n(d\varepsilon n,(1-\varepsilon-\delta)n,cn) = C^{cn}_{(1-\varepsilon-\delta)n} \sum_{k=0}^{cn} (-1)^k C^k_{cn} \frac{\left(\varepsilon+\delta+c-\frac{k}{n}\right)^{d\varepsilon n}}{1+\frac{k}{(1-\varepsilon-\delta-c)n}}$$

В результате получена точная формула для величины P. Её можно доказать и непосредственно с помощью прямой подстановки. Правда, в этом случае будет не ясен путь её получения.

Оценим сверху |P|:

$$|P| \le C^{\varepsilon n}_n C^{\delta n}_{(1-\varepsilon)n} C^{cn}_{(1-\varepsilon-\delta)n} (cn+1) 2^{cn} (\varepsilon+\delta+c)^{d\varepsilon n} \mapsto 0$$

Выполним разложение полученной оценки по формуле Стирлинга:

$$C^r_k = \frac{k!}{r!(k-r)!} < \exp\{k\ln k + \tfrac{1}{2}\ln k - r\ln r -$$
$$-\tfrac{1}{2}\ln r - (k-r)\ln(k-r) - \tfrac{1}{2}\ln(k-r)\}$$

и раскроем все скобки:

$$P < C_n^{\varepsilon n} C_{n-\varepsilon n}^{a\varepsilon n} n C_{d\varepsilon n}^{b\varepsilon n} (3\varepsilon + a\varepsilon)^{d\varepsilon n - b\varepsilon n} <$$
$$< \exp\{[n\ln n + \tfrac{1}{2}\ln n - \varepsilon n \ln \varepsilon n - \tfrac{1}{2}\ln \varepsilon n -$$
$$-(n-\varepsilon n)\ln(n-\varepsilon n) - \tfrac{1}{2}\ln(n-\varepsilon n)] + [(n-\varepsilon n)\ln(n-\varepsilon n) +$$
$$+\tfrac{1}{2}\ln(n-\varepsilon n) - a\varepsilon n \ln a\varepsilon n - \tfrac{1}{2}\ln a\varepsilon n - (n-\varepsilon n - a\varepsilon n)\ln(n-\varepsilon n - a\varepsilon n) -$$
$$-\tfrac{1}{2}\ln(n-\varepsilon n - a\varepsilon n)] + \ln n + [d\varepsilon n \ln d\varepsilon n + \tfrac{1}{2}\ln d\varepsilon n - b\varepsilon n \ln b\varepsilon n -$$
$$-\tfrac{1}{2}\ln b\varepsilon n - (d\varepsilon n - b\varepsilon n)\ln(d\varepsilon n - b\varepsilon n) - \tfrac{1}{2}\ln(d\varepsilon n - b\varepsilon n)] +$$
$$(d\varepsilon n - b\varepsilon n)\ln(3\varepsilon + a\varepsilon)\} =$$
$$= \exp\{n[\varepsilon(d-1-a-b)\ln\varepsilon - \varepsilon a \ln a + 2\varepsilon \ln 2 - \varepsilon b \ln b -$$
$$-\varepsilon(2-b)\ln(2-b) + \varepsilon(2-b)\ln(3+a) - (d-1-\varepsilon-\varepsilon a)\ln(d-1-\varepsilon-\varepsilon a)] +$$
$$+[-\tfrac{1}{2}\ln n - \tfrac{1}{2}\ln\varepsilon - \tfrac{1}{2}\ln a - \tfrac{1}{2}\ln\varepsilon - \tfrac{1}{2}\ln(d-1-\varepsilon-\varepsilon a) + \tfrac{1}{2}\ln 2 + \tfrac{1}{2}\ln\varepsilon -$$
$$-\tfrac{1}{2}\ln b - \tfrac{1}{2}\ln\varepsilon - \tfrac{1}{2}\ln\varepsilon - \tfrac{1}{2}\ln(2-b)]\} <$$
$$< \exp\{n[\varepsilon(d-1-a-b)\ln\varepsilon - \varepsilon a\ln a + \varepsilon\ln 4 - \varepsilon b\ln b - \varepsilon(2-b)\ln(2-b) +$$
$$+\varepsilon(2-b)\ln(3+a) - (d-1-\varepsilon-\varepsilon a)\ln(d-1-\varepsilon-\varepsilon a)]\} <$$
$$\exp\{n[\varepsilon[-(d-1-a-b)\,|\ln\varepsilon\,| - a\ln a + 2\ln 2 - b\ln b - (2-b)\ln(2-b) +$$
$$+(2-b)\ln(3+a)] + |\ln(d-1-\varepsilon(1-a))|]\}$$

Пусть $a_0 > 0$ и $b > 0$ зафиксированы произвольным образом, но так, что $a_0 + b < 1$. Теперь учтём, что при $\varepsilon \to 0$ $|\ln(1-\varepsilon(1+a))| \to \varepsilon(1+a)$ и $|\ln\varepsilon| \to \infty$. Поэтому найдётся такое ε_0, при котором показатель в последнем выражении будет меньше некоторой отрицательной константы. Это и будет означать, что $P(\varepsilon, a\varepsilon, b\varepsilon, n) \to 0$ при $n \to \infty$ для любых $\varepsilon \le \varepsilon_0$ и $a \le a_0$. Теорема доказана.

Заключение

В данной работе были приведены анализ и методы решения нескольких задач.

Для первой задачи (задачи верификации TCTL) был построен алгоритм, который поддерживает широкий класс *TCTL*-формул и строит по ним регионные автоматы. Размеры полученных автоматов являются достаточно малыми и не изменяются при изменении всех таймеров в константу раз. Одна из верхних оценок на число состояний регионного автомата имеет следующее достоинство: если в алгоритме, изложенном в работе [2], это число всегда было пропорционально объёму фазовой области, то в данной работе оно (при определённых условиях) ограничено лишь площадью её поверхности (точнее, площадью её проекции вдоль оси времени). Построенный алгоритм был протестирован на системе управление железнодорожным переездом, а также на автоматной задаче про туристов и сломанный мост [2]. Для этих задач данный алгоритм конструирует регионный автомат, размер пространства состояний которого приблизительно в 25 раз меньше, чем для алгоритма, описанного в работе [2].

Также в работе (вторая задача) выполнено сведение задачи *SAT* к задаче 1 - *MINESWP* (сведение выполнялось с целью решать задачу 1 - *MINESWP* динамикой по профилю, но попутно оно показывает *NP*-полноту этой задачи). До этого было известно лишь сведение к более общей задаче 8 - *MINESWP* [20].

Время работы динамики по профилю экспоненциально не по k-мерном объёму игрового поля, а по площади его гиперграни (по (k–1)-мерной площади поверхности).

Следующий результат относится к оценкам на минимальный объём реализации n-вершинного графа в k-мерном пространстве. В работе получена нижняя асимптотическая оценка на этот объём $\Omega\left(n^{\frac{1}{k-1}}\right)$, которая выполняется как в худшем, так и в среднем случае. Она указана для суммарного объёма несмежных рёбер этого графа, если считать их проволоками фиксированной толщины. Геометрически это может быть проиллюстрировано следующим образом. Пусть имеется k-мерный контейнер, целиком заполненный жидкостью. При погружении графа ограниченной смежности с n вершинами в этот контейнер часть жидкости вытесняется. Можно сказать, что нижняя оценка получена на минимальный объём вытесненной доли этой жидкости. Эти оценки означают, что при реализации графа в k-мерной области число его вершин и рёбер пропорционально не объёму этой области, а площади её поверхности.

Также в работе была получена формула в явном виде для определения я числа универсальных графов, из которой следует, что наугад взятый граф является универсальным с вероятностью, стремящейся к единице. Для доказательства формулы решалось рекуррентное (разностное) уравнение в частных разностях в трёхмерном дискретном фазовом пространстве. При

решении этого уравнения применялось понижение размерности фазового пространства: вначале путём замены оно было преобразовано к системе (семейству) уравнений на двумерном фазовом пространстве, а затем каждое из полученных уравнений методом Буля было декомпозировано на последовательность операторных уравнений в одномерном фазовом пространстве, которые решались непосредственно.

Изложенное позволяет утверждать, что для многих задач, связанных с областями многомерного пространства (в частности, дискретного пространства), можно применять методы понижения размерности, которые сводят рассмотрение k-мерного тела к рассмотрению его (k–1)-мерной поверхности. Эти методы ранее применялись к операционному исчислению, а в данной работе были применены к задаче верификации TCTL, комбинаторной задаче на игровом поле (точнее, выполнению динамики по профилю для этой игры) и задаче оптимальной укладки графов в многомерном пространстве. В результате был получен эффективный способ представления регионного автомата, алгоритм решения игровой задачи и нижняя оценка для оптимальной укладки графов.

Источники

1. *Кларк Э. М., Грамберг О., Пелед Д.* Верификация моделей программ: Model checking. М.: МНЦМО, 2002.
2. *Katoen J.-P.* Concepts, Algorithms, and Tools for Model Checking. Lehrstuhl für Informatik VII, Friedrich-Alexander Universität Erlangen-Nürnberg //Lecture Notes of the Course «Mechanised Validation of Parallel Systems» (course number 10359). Semester 1998/1999.
3. *D'Argeno P. R., Katoen J.-P., Ruys T., Tretmans G.-J.* The bounded retransmission protocol must be on time! //Tools and Algorithms for the Construction and Analysis of Systems. LNCS 1217. 1997, pp. 416–432.
4. *Abdulla P. A., Mahata P., Mayr R.* Dense-Timed Petri Nets: Checking Zenoness, Token Liveness and Boundedness //LMCS. Vol. 3 (1:1). 2007, pp. 1–61. http://www.lcms-online.org
5. *Möller M. O.* Structure and Hierarchy in Real-Time Systems. PhD Dissertation. BRICS PhD School. Department of CS, University of Aarhus. Denmark, 2002.
6. *Rokicki T. G., Myers C. J.* Automatic Verification of Timed Circuits. Stanford University. Stanford, CA 94305.
7. *Larsen K. G., Peterson P., Yi W.* Model-checking for Real-Time Systems.
8. *Alur R, Henzinger T. A.* Back to the future: towards a theory of timed regular languages / IEEE Symp. on Foundations of CS. 1992, pp. 177–186.
9. *Alur R, Henzinger T. A.* Real-time logics: Complexity and expressiveness // Information and Computation. 104: 35–77. 1993.
10. *Alur R., Courcoubetis C., Dill D.* Model-checking in dense real-time // Information and Computation, 104: 2–34, 1993.
11. *Alur R., Dill D.* Automata-theoretic verification of real-time systems // Formal Methods for Real-Time Computing. 1996, pp. 55–82.

12. *Alur R., Courcoubetis C., Halbwachs N., Dill D., Wong-Toi H.* Minimisation of Timed Transition Systems. http://cs-structure.inr.ac.ru
13. *Alur R., Courcoubetis C., Dill D.* Model-checking for Probabilistic real-time systems / 18th ICALP. LNCS 510, 1991. http://cs-structure.inr.ac.ru
14. *Alur R., Courcoubetis C., Dill D.* Verifying Automata Specifications of Probabilistic Real-Time Systems. http://cs-structure.inr.ac.ru
15. *Alur R., Courcoubetis C., Halbwachs N., Dill D., Wong-Toi H.* An implementation of Three Algorithms for Timing Verification Based on Automata Emptiness. http://cs-structure.inr.ac.ru
16. *Courcoubetis C., Alur R., Dill D., Yannakakis M.* Topics in Probabilistic Verification. http://cs-structure.inr.ac.ru
17. *Alur R., Dill D.* A Theory of Timed Automata / 17th International Colloquium on Automata, Languages and Programming (1990), REX Workshop «Real-Time: theory in practice» (1991).
18. *Fraenkel A. S.* Combinatorial Games: Selected Bibliography with a Succint Gourmet Introduction. http://www.combinatorics.org/Surveys/ds2.pdf
19. *Minesweeper is NP-complete.* http://sed.free.fr/complex/mines.html
20. *Kaye R.* How complicated is Minesweeper? // ASE 2003. http://web.mat.bham.ac.uk/R.W.Kaye
21. *Camargo R. S.* A possible explanation for the Board Cycle Bug of Microsoft Minesweeper. http://planet-minesweeper.com
22. *Kadlac M., Cull P.* Explorations of the Minesweeper Consistency Problem, 2003.
23. *Stewart I.* Million-Dollar Minesweeper, 2000. http://www.minesweeper.info/articles/
24. *Collet R.* Playing the Minsweeper with Constraints.
25. *Copp H.* Truffle-swine keeper, 2004. http://people.freenet.de/hskopp/swinekeeper.html
26. *Studholme C.* Minsweeper as a Constraint Satisfaction Problem.

27. *Kaye R.* Infinite versions of Minesweeper are Turing-complete, 2007.
28. *Castillo L. P., Wrobel S.* Learning Minesweeper with Multirelational Learning.
29. *Kaye R.* Some Minesweeper Configurations, 2007.
30. *Minesweeper and Propositional Logics* / R 2004: Lab session, February 6.
31. *Pedersen K.* The complexity of Minesweeper and game playing strategies, 2004.
32. *Fix J. D, McPhail B.* Offline 1-Minesweeper is NP-complete, 2004.
33. *Негодаев М. А.* Проблема выполнимости линейной карты для игр «Сапёр» и «Цветной Сапёр», 2006. http://proceedings.usu.ru/mag/0043(05_01-2006)/a05.ps
34. *Негодаев М. А.* Вычислительная сложность инверсных проблем «Сапёр» и «Цветной Сапёр». http://kungurka.imm.uran.ru/inf/sbornik07/kung07s377.pdf
35. *Негодаев М. А.* Задача MINESWEEPER и приближённые алгоритмы её решения. http://kungurka.imm.uran.ru/inf/history37/sbornik/sbornik.php
36. *Stuckman J., Zhang G.* Mastermind is NP-complete. arXiv:cs.CC/0512049v1, 2005.
37. *Cormode G.* The Hardness of the Lemmings Game, or “Or no, more NP-completeness Proofs” / DIMACS Technical Report 2004–11.
38. *Andersson D.* HIROIMONO is NP-complete // BRICS RS-07-1, 2007.
39. *Friedman E.* The Game of Cubic is NP-complete.
40. *Hearn R. A.* TipOver is NP-complete.
41. *Дворкин М. Э.* Методы автоматизации доказательства NP-полноты двумерных локально-зависимых задач. Бакалаврская работа. СПбГУ ИТМО, 2008. http://rain.ifmo.ru/~dvorkin/science/bachelors.pdf
42. *Колмогоров А. Н., Барздинь Я. М.* О реализации сетей в трёхмерном пространстве // Проблемы кибернетики. Вып. 19, с. 261 – 268.
43. *Rosenberg A. L.* Three-Dimensional VLSI: A Case Study //Math. Systems Theory 16 (1983), 1–8.

44. *Касьянов В. Н., Евстигнеев В. А.* Графы в программировании: обработка, визуализация и применение. СПб.: БХВ-Петербург, 2003.
45. *Зыков А. А.* Основы теории графов. М.: Наука, 1987.
46. *Brandenburg F. J.* Graph Drawing: past–present–future, 2002.
47. *Eades P., Symvonis A., Whitesides S.* Three-dimensional orthogonal graph drawing algorithms, 1999.
48. *Eades P., Stirk C., Whitesides S.* The Techniques of Kolmogorov and Barzdin for Three Dimensional Orthogonal Graph Drawings. Technical Report 95–07. Department of CS. University of Newcastle. Newcastle NSW 2308 Australia, 1995. http://cs-structure.inr.ac.ru
49. *Eades P., Symvonis A., Whitesides S.* Two Algorithms for Three Dimensional Orthogonal Graph Drawing, 2006. http://cs-structure.inr.ac.ru
50. *Biedl T., Chan T. M.* A note on 3D orthogonal graph drawing. DAM, 2005.
51. *Papakostas A., Six J. M., Tollis I. G.* Experimental and Theoretical Results in Interactive Orthogonal Graph Drawing. Department of CS, The University of Texas at Dallas. Richardson, TX 75083-0688, 1996. http://cs-structure.inr.ac.ru
52. *Biedl T., Shermer T., Whitesides S., Wismath S.* Bounds for Orthogonal 3-D Graph Drawing // Journal of Graph Algorithms and Applications. Vol. 3, 1999. no. 4, pp. 63 – 79. http://www.cs.brown.edu/publications/jgaa/
53. *Papakostas A., Tollis I. G.* Algorithms for Incremental Orthogonal Graph Drawing in Three Dimensions // Journal of Graph Algorithms and Applications/ Vol. 3. 1999. no. 4, pp. 81–115. http://www.cs.brown.edu/publications/jgaa/
54. *Di Battista G., Patrignani M.* A Split&Push Approach to 3D Orthogonal Drawing // Journal of Graph Algorithms and Applications. Vol. 4. 2000. no. 3, pp. 105–133. http://www.cs.brown.edu/publications/jgaa/

55. *Wood D. R.* Lower Bounds for Number of Bends in Three-Dimensional Orthogonal Graph Drawings // Journal of Graph Algorithms and Applications/ Vol. 7. 2003. no. 1, pp. 33–77. http://jgaa.info
56. *Harel D., Koren Y.* Graph Drawing by Higher-Dimensional Embedding // Journal of Graph Algorithms and Applications. Vol. 8. 2004. no. 2, pp. 195–214. http://jgaa.info
57. *Eades P., Healy P.* Special Issue on Selected Papers from the Thirteenth International Symposium on Graph Drawing, GD 2005: Guest Editor's Foreword // Journal of Graph Algorithms and Applications. Vol. 11. 2007. no. 2, pp. 323, 324. http://jgaa.info
58. *Brandes U., Fleischer D., Puppe T.* Dynamic Spectral Layout with an Application to Small Worlds // Journal of Graph Algorithms and Applications. Vol. 11. 2007. no. 2, pp. 325–343. http://jgaa.info
59. *Hachul S., Jünger M.* Large-Graph Layout Algorithms at Work: An Experimental Study // Journal of Graph Algorithms and Applications. Vol. 11. 2007. no. 2, pp. 345–369. http://jgaa.info
60. *Hong S., Nikolov N. S., Tarassov A.* A 2.5D Drawing of Directed Graphs // Journal of Graph Algorithms and Applications. Vol. 11. 2007. no. 2, pp. 371–396. http://jgaa.info
61. *Huang W., Hong S., Eades P.* Effects of Sociogram Drawing Conventions and Edge Crossings in Social Network Visualization // Journal of Graph Algorithms and Applications. Vol. 11. 2007. no. 2, pp. 397–429. http://jgaa.info
62. *Lin C., Yen H.* On Balloon Drawings of Rooted Trees // Journal of Graph Algorithms and Applications. Vol. 11. 2007. no. 2, pp. 431–452. http://jgaa.info
63. *Noack A.* Energy Models for Graph Clustering // Journal of Graph Algorithms and Applications. Vol. 11. 2007. no. 2, pp. 453–480. http://jgaa.info

64. *Papadopoulos C., Voglis C.* Drawing graphs using modular decomposition // Journal of Graph Algorithms and Applications. Vol. 11. 2007. no. 2, pp. 481–511. http://jgaa.info
65. *Dujmović V.* Track Layouts of Graphs. PhD thesis. School of CS. McGill University. Montréal, 2003.
66. *Dujmović V., Wood D. R.* Three-Dimensional Grid Drawings with Sub-quadratic Volume.
67. *Dujmović V., Fellows M., Hallett M., Kitching M., Liotta G., McCartin C., Nishimura N., Radge P., Rosamond F., Suderman M., Whitesides S., Wood D. R.* On the Parameterized Complexity of Layered Graph Drawing, 2001. http://cs-structure.inr.ac.ru
68. *Hanlon S.* Visualizing Three-dimensional Graph Drawings. Department of Mathematics and CS. University of Lethbridge, 2006.
69. *Chrobak M., Goodrich M. T., Tamassia R.* On the Volume and Resolution of 3-Dimensional Convex Graph Drawing. Extended Abstract. http://cs-structure.inr.ac.ru
70. *Wood D. R.* An Algorithm for Three-Dimensional Orthogonal Graph Drawing. School of Computer Science and Software Engineering Monash University. Clayton, VIC 3168. Australia, 1998. http://cs-structure.inr.ac.ru
71. *Wood D. R.* A New Algorithm and Open Problems in Three-Dimensional Orthogonal Graph Drawing. School of Computer Science and Software Engineering Monash University. Clayton. VIC 3168, Australia. http://cs-structure.inr.ac.ru
72. *Wood D. R.* On Higher-Dimensional Orthogonal Graph Drawing. School of Computer Science and and Software Engineering Monash University Wellington Road, Clayton. VIC 3168. Australia, 1996. http://cs-structure.inr.ac.ru
73. *Wood D. R.* Multi-Dimensional Orthogonal Graph Drawing in the General Position Model. Technical Report 1999/38, School of Computer Science and

Software Engineering Monash University. Clayton. VIC 3168, Australia. http://cs-structure.inr.ac.ru

74. *Wood D. R.* Drawing a Graph in a Hypercube. School of Computer Science and School of Mathematics and Statistics. McGill University. Montréal. Canada, 2004. http://cs-structure.inr.ac.ru
75. *Biedl T., Thiele T., Wood D. R.* Three-Dimensional Orthogonal Graph Drawing with Optimal Volume / 8th International Symposium on Graph Drawing (GD 2000). Colonial Williamsburg. Virginia. USA, 2000. http://cs-structure.inr.ac.ru
76. *Gajer P., Kobourov S. G.* GRIP: Graph dRawing with Intelligent Placement //GD 2000, LNCS 1984. 2001, pp. 222–228, Springer-Verlag Berlin Heidelberg.
77. *Hernando C., Mora M., Pelayo I. M., Seara C., Wood D. R.* Extremal Graph Theory for Metric Dimension and Diameter. arXiv:0705.0938v1 [math.CO], 2007.
78. *Pajntar B.* Overview of Algorithms for Graph Drawing. Department of Knoledge Technologies. Jozef Stefan Institute.
79. *Wood D. R.* Degree constrained book embeddings // Journal of Algorithms. 45 (2002), pp. 144–154.
80. *Wood D. R.* Optimal three-dimensional orthogonal graph drawing in the general position model // Theoretical CS. 299 (2003), pp. 151–178.
81. *Wood D. R.* Minimizing the Number of Bends and Volume in Three-Dimensional Orthogonal Graph Drawings with a Diagonal Vertex Layout. AWOCA'97. Noosa. Queensland. Australia, 2001. http://cs-structure.inr.ac.ru
82. *Wood D. R.* Multi-Dimensional Orthogonal Graph Drawing with Small Boxes. School of Computer Science and Software Engineering Monash University. Melbourne. Australia. http://cs-structure.inr.ac.ru
83. *Wood D. R.* Bounded Degree Book Embeddings and Three-Dimensional Orthogonal Graph Drawing. Extended Abstract. Basser Department of Computer

Science. The University of Sydney. Sydney NSW. Australia, 2006. http://cs-structure.inr.ac.ru

84. *Wood D. R.* Three-Dimensional Orthogonal Graph Drawing. PhD thesis. School of Computer Science and Software Engineering Monash University, 2000. http://cs-structure.inr.ac.ru
85. *Jagerman D. L.* Difference equations with applications to queues. N.Y.: Marcel Dekker, Inc. RUTCOR, 2000.
86. *Jordan C.* Calculus of Finite Differences, Second Edition. N.Y.: Chelsea Publishing Company, 1950.
87. *Boole G.* A treatise on the Calculus of Finite Differences. Cambridge, 1860.