ків так це і роблять. Недарма в стандарті SQL з'явилися спеціальні типи даних *DATE* і *TIME*. Але в такому підході закладено декілька вад:

- СКБД не знає семантики часового поля відношення і не може контролювати коректність його значень;
- з'являється додаткова збитковість схову — попередній стан об'єкта даних зберігається і в основній БД, і в часописі змін;
- мови запитів реляційних СКБД не пристосовані для роботи з часом.

Існує окремий напрямок досліджень і розробок в області темпоральних БД. У цій області досліджуються питання моделювання даних, мови запитів, організація даних у зовнішній пам'яті тощо. Основна теза темпоральних систем полягає в тому, що для будь-якого об'єкта даних, створеного в момент часу t_1 і знищеного в момент часу t_2, у БД зберігаються, і є доступними для користувачів, усі його стани в часовому інтервалі $[t_1, t_2]$.

Дослідження і побудови прототипів темпоральних СКБД звичайно виконуються на основі деякої реляційної СКБД. Як і у випадку дедуктивних БД, темпоральна СКБД — це надбудова над реляційною системою. Звичайно, це не кращий спосіб реалізації з погляду ефективності, але він простий і дозволяє виконувати досить глибокі дослідження.

У доповіді викладаються питання побудови постреляційної системи з темпоральними можливостями. Характерні особливості системи, що пропонується, полягають у такому:

- система керування пам'яттю підтримує історичні дані;
- запити можуть містити часові характеристики об'єктів.

ДОСЛІДЖЕННЯ СИСТЕМИ ВЗАЄМОДІЮЧИХ ПРОЦЕСІВ ЗА ДОПОМОГОЮ ТЕОРІЇ АВТОМАТІВ

М. Глибовець (кафедра інформатики)

Для аналізу функціювання складної мережі найважливішими є два класичних поняття: визначеності (досягнення, живучості) і блокування (жоден процес не може працювати). Проміжний стан мережі можна трактувати як певний стан керування автомату. Перехід зміни стану мережі приводить до зміни стану керування автомату. Перевагою такого підходу є можливість використання єдиного підходу для моделювання програм, процесів, які вони описують і способи взаємодії цих процесів. Програми будемо розглядати на рівні блок-схем. Враховуючи паралельний характер роботи мережі, ми

зупинимося на трьох основних проблемах: незалежні завдання (задачі, протоколи) не повинні зіткнутися в системі; процеси, які взаємодіють між собою, мають це робити в заздалегідь визначеному порядку; блокування. Вирішення перших двох задач об'єднаємо в одну задачу, яку будемо називати визначеністю. Процес будемо описувати автоматом, який не має обмежень як на множину станів керування, так і на множину правил переходів. Послідовні взаємодіючі процеси з'єднуються в єдиний процес за допомогою структурування станів у вигляді декартового добутку компонент, які відповідають коміркам програмної пам'яті, гомоморфізму процесів та зведенню інформації про зміну стану до простих операцій, котрі будуть використовуватись у вигляді міток переходів. Цю модель можна розглядати як спробу формалізації моделі Дейкстри "взаємодіючих процесів".

Послідовні процеси будемо задавати впорядкованою парою $P=(Q,R)$, де Q — не пуста множина станів, R — бінарне відношення на множині Q (переходи).

Процес можна проінтерпретувати як орієнтований граф класичним чином: вершини графу будуть відповідати станам керування, а ребра — можливим переходам. Термінологію процесу зручніше використовувати при задані динамічної інтерпретації, а термінологію графу — для статичного сприйняття.

Процеси, які визначаються машинними програмами, змінюють свій стан, виконуючи інструкції програми. Тому це можна зафіксувати, ввівши розмітку ребер графу станів символьними знаками відповідних інстпукцій програми. Нехай L буде визначати цю множину інструкцій програми, тоді визначення R можна розширити до $R \subseteq Q \times L \times Q$ так, що (a,s,b)$\in R$ буде позначати факт, що на множині переходів є перехід (ребро) помічений s, який з'єднує стан a зі станом b.

Формалізуємо поняття обчислення. Нехай у графі станів маємо шлях: (a_0, s_1, a_1), (a_1, s_2, a_2), …,(a_{n-1}, s_n, a_n) який з'єднує початковий та кінцевий стан процесу. Тоді проміжні вершини шляху будуть відповідати можливим проміжним обчисленням процесу. Якщо такий шлях існує, тоді стан a_0 називають досяженим із a_n. Домовимось також, що кожна вершина може бути досягнута із самої себе з довжиною шляху, рівною нулеві.

Послідовність операцій $s_1, s_2, \ldots, s_n$ вздовж шляху називається слідом. Множина всіх слідів, які визначаються графом, називається мовою. Тип мови визначає тип графу. Наприклад, якщо граф скінчений, тоді мова буде регулярною.

Завдання обчислення в вигляді шляху відображає дискретну природу цифрових обчислень. Тому має значення тільки впорядкування станів під час обчислень.

Аналіз роботи процесів вимагає, щоб після обробки граф залишався без змін для подальшого аналізу. Деякі програми мають нескінченні графи станів, які повинні бути представлені в скінченому варіанті. Традиційно структурно-зберігаючі перетворення даних називають гомоморфізмом. В графі станів зберігаючою структурою буде множина ребер.

Нехай $P_1 = (Q_1, R_1)$, $P_2 = (Q_2, R_2)$, і $F: Q_1 -> Q_2$ буде перетворенням даних. Тоді F буде гомоморфізмом тоді і тільки тоді, коли $(a,s,b) \in R_1$ буде визначати що $(f(a),s,f(b)) \in R_2$.

Згорткою називається гомоморфізм з додатковою властивістю:

коли $(a', s, b') \in R_2$, тоді буде існувати таке $(a,s,b) O R_1$, що $a'=f(a)$, а $b'=f(b)$.

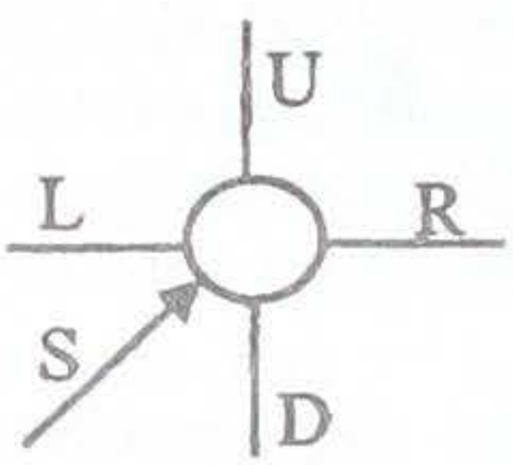
Ізоморфізм це гомоморфізм, інверсія якого також буде гомоморфізмом. Відмітимо, що ізоморфізм буде також згорткою.

Має місце таке твердження: Гомоморфізм зберігає сліди.

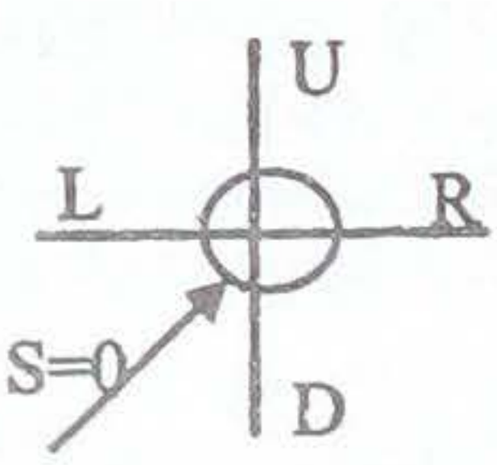
СХЕМИ НА КОМУТАЦІЙНИХ ЕЛЕМЕНТАХ

М. Глибовець (кафедра інформатики НаУКМА)
М. Медведєв (Національний університет ім. Т. Шевченка)

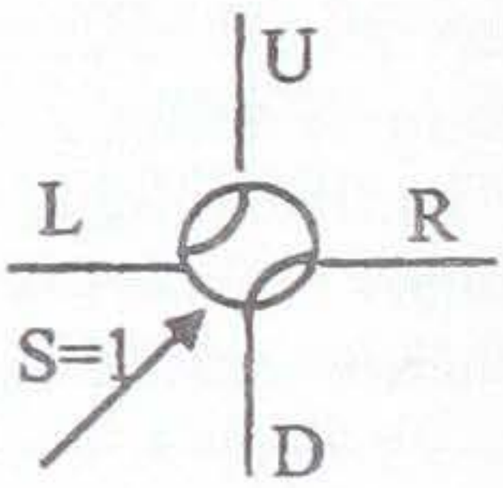
Комутаційним елементом (далі — КЕ) називатимемо елемент, який зображено на малюнку 1. Він має п'ять полюсів, чотири з яких (L,R,U,D) є інформаційними, а п'ятий (S) — керуючим. В залежності від сигналу на вході S (0 або 1) КЕ може знаходитися в одному з двох станів, які зображено на малюнках 2, 3.



Мал. 1.



Мал. 2.



Мал. 3.