Announcements

Project #2 will be posted today.

Friday (next) lecture:
  → Shaleen will talk about polyhedral optimizations
  → Phillip will present work on intermittent energy optimizations

Papers are posted under sakai/resources

I will send out emails to each of you asking to sign up for particular papers.
KDG (knob dependence graph) is an RSDG without redundancy

**Developer**

- Weighted KDG specification
  - Discrete node levels
  - AND dependencies only
  - No redundancy

**User**

- Weighted KDG in xml format
  - Quality and cost weights

**Project Code to be written by students**

- Read in KDG
- Read in budget
- Generate 0-1 problem

**C++ program**

- Execution time budget (command line argument)

**0-1 integer program KDG.lp**

**Gurobi solver**

- Optimal configuration
fully weighted KDG: cost and quality

KDG.desc:

<Knobs>
// (cost-qual)
S1 [(8-5), (10-21), (15-40), (20-50)]
S2 [(20-10), (25-20)]
</Knobs>

<KDG>  

S1 S2
  50 25
  40 20
  21
  8

S1.0 S1.1 S1.2 S1.3 S2.0 S2.1

KDG.lp with budget=70:

Maximize
20 S1.3 + 15 S1.2 + 10 S1.1 + 5 S1.0 +
20 S2.1 + 10 S2.0
Subject To
cost: 50 S1.3 + 40 S1.2 + 21 S1.1 + 8 S1.0 +
25 S2.1 + 20 S2.0 <= 70
S1.Select: S1.0 + S1.1 + S1.2 + S1.3 = 1
S2.Select: S2.0 + S2.1 = 1
Binary
S1.0 S1.1 S1.2 S1.3 S2.0 S2.1

End

Project Code to be written by students

parse.py

Weighted KDG in xml format

gurobi

Objective
val = 35
S1.2 = 1
S2.1 = 1

Project #2 - Sample KDG

cs516, spring 20 © U.Kremer
The second law of thermodynamics: Total entropy of an isolated system can never decrease over time, and is constant if and only if all processes are reversible.

Losing information from a digital system by erasing/overwriting bits necessarily implies ejecting that information into the system’s environment.

Once “absorbed” into the environment, information that was previously known (correlated) becomes entropy, i.e., unknown / uncorrelated information.
Irreversible systems lose information which is directly related to dissipating energy into the environment (increasing entropy). Example: irreversible: drop a ball - it will end up on the ground, but from what height was it thrown? reversible: ideal situation without friction - will return to the height it was dropped from.

information loss == energy loss, i.e., deleting information “costs” / dissipates energy
Why look at reversible computing?

Many basic physical laws are reversible, i.e., can be applied forwards and backwards in time.

Pendulum example

If you know the position and velocity of the pendulum at time $t_0$, do you know its position and velocity at time
- $t_0 + 5$ secs?
- $t_0 - 5$ secs?
How does irreversible computing relate to current computing?

Current computation use CMOS where erasing a bit leads to power dissipation (registers, caches, memory, ...). The final answer of the computation typically does not let us determine its input in a deterministic way.

Example: Observation: Your program prints '5'
Question: What was the input for your program?

Current computation is forward computation only. Reversible computing includes forward and backward computation.
Theoretical benefits: No information is lost in the system, which means that no energy is dissipated.

Noise to signal ratios: Irreversible systems have a theoretical fixed limit on how much power must be dissipated for a processing a piece of information (e.g., a bit switch). This is called the Landauer’s limit: $k_B T \ln 2$ ($k_B$ - Boltzmann constant, $T$ - temperature) and results in 1 electron volt per bit operation energy cost at room temperature (20°C).

Current technology is FAR away from this limit, but the limit is real and holds for ALL computing technologies.

Reversible computing does not have such a theoretical limit.
International Technology Roadmap for Semiconductors (ITRS)

Why look at reversible computing?

The “Forever Forbidden Zone” for All Irreversible Computing

Any Hope of Sustained Long-Term Progress Absolutely Requires Reversible Computing!
Potential benefits of reversible computing

- Less energy consumption (minimal energy leakage), with close to 0 energy limit
- Less thermal dissipation / no cooling requirements

We could build large, compact computers with many layers, small energy supplies, and no or only minimal need for cooling!
But how practical is reversible computing?

- Quantum computing is reversible computing; in contrast to reversible computing, quantum computing is not a general purpose computing paradigm, i.e., quantum works well only for specific classes of problems.

- People have explored the ideas of reversible computing with some success:
  - Adiabatic logic/circuits
  - Nano-technology devices
  - Ballistic systems where synchronized data signals self-propagate through the device and initiate transitions (no outside clock)
  - Reversible touring machines (theoretical results)

As with quantum computing, much more research is needed to produce devices and software tool chains to make reversible computing practical.
Note of caution:
Fundamental physical properties/insights may not directly translate into effective engineering solutions!

However, they provide a limit to how far we can go. Substantial R&D is needed to make reversible computing a practical reality.
Things we want to cover

(1) How to design circuits that are reversible?

(2) Can we design programming languages that are reversible?
   - Reversible programs need to run on reversible hardware
   - Janus programming language

(3) How to write programs / implement algorithms in such a reversible language?
A gate is **logically reversible** if its input value(s) can be uniquely determined from its output value(s) and vice versa. In other words, there is a **one-to-one mapping** between input and output vectors:

\[ F(x) = F(y) \implies x = y \]

Which of these gates are reversible?
A gate is **logically reversible** if its input value(s) can be uniquely determined from its output value(s) and visa versa. In other words, there is a **one-to-one mapping** between input and output vectors:

\[ F(x) = F(y) \Rightarrow x = y \]

Which of these gates are reversible?
How to construct a reversible XOR and AND gates?

Reversible gates:
- Same number of inputs and outputs
- Need to introduce “garbage” inputs/outputs

There are automatic technique to synthesize reversible circuits from irreversible circuit specifications (e.g.: using Toffoli gates)
“Adiabatic”: Total heat / energy in a system remains the same

Reversible circuits are implemented using adiabatic logic components, using **two rules for adiabatic circuits**: (1) Never turn on a transistor if there is a voltage difference between the drain and source; (2) Never turn off a transistor that has current flowing through it

**Example:**

**Conventional vs. Adiabatic Charging (CMOS)**

- **Conventional charging:**
  - Constant voltage source
  - Energy dissipated: \( E_{\text{conv}} = \frac{1}{2} CV^2 \)

- **Ideal adiabatic charging:**
  - Constant current source
  - Energy dissipated: \( E_{\text{adia}} = \frac{Q^2 R}{2t} = CV^2 \frac{RC}{2} \)

**Note:** Adiabatic charging beats the energy efficiency of conventional by advantage factor: \( A = \frac{E_{\text{adia}}}{E_{\text{conv}}\text{diss}} = \frac{1}{2} \frac{t}{RC} \)
There are adiabatic system implementations (e.g.: MIT Pendulum project 1997-99); however, mainly proof-of-concept.

Challenges for CMOS:
- Need for precise trapezoidal waveform to drive the charging and discharging process in CMOS circuit;
- Current adiabatic circuits are typically much slower than their traditional logic counterparts

A lot more R&D needs to be done to make this technology more effective; other technologies in addition to CMOS: adiabatic superconducting logic, nano-mechanical rod logic, quantum-dot cellular automata, ...
Can we design programming languages that are reversible?
   - Example: Janus programming language

Janus online interpreter:
http://topps.diku.dk/pirc/?id=janusP