NWI-IBC019: Operating systems

Bernard van Gastel and Nils Jansen

I/O Systems
UPPAAL Assignment

Deadline: **Sunday, November 12, 2017** (more than week after the exam)

Go to the practicals!

Use UPAAL 4.0 (VM)

Start now!

Keep page limits in the report!
Theme: Verified Machine Learning

If you are interested in any of the topics listed below or in the general area, please contact me (Nils Jansen). My webpage can be found here.

With rapidly growing application of artificial intelligence and machine learning in social life, considering for instance autonomous systems such as self-driving cars, the need for verified guarantees against potentially fatal accidents is self-evident. An important question is then, how confidence in system behaviors obtained from machine learning can be transferred to formal verification.

Vice versa, industrial usage of formal verification so far suffers from the fact that scalability is still an issue for large real-world applications. Leveraging the capabilities of machine learning to assess large data sets will help to enable verification for more realistic systems.

A very nice article discussing challenges and open problems can be found here. If one of the open problems is of interest, let me know. On top of that, please consider the following concrete thesis proposals. We focus on two fundamental formalisms/methods, namely partially observable Markov decision processes (POMDPs) and reinforcement learning (RL). If interest in deep neural networks (DNNS) is there, consider reading this article. I’m happy to discuss potential projects.

Thesis Topics

a) Data-consistent Machine Learning with Formal Guarantees

In machine learning algorithms, a system model is learned based on observation of the real world. Upon further observation, this model may be subject to change. The problem of applying formal verification to such a model is that it is not a well-defined and fixed model of the system at hand. This project proposes to robustify a current learned model against further changes based on future observations. If it can be verified, that given
I/O Systems

Structure of I/O subsystem
Principles of I/O hardware
Performance Aspects
I/O Hardware

- Large **variety** of I/O devices
  - Storage
  - Transmission
  - Human-interface
I/O Hardware

- **Concepts** – signals from I/O devices
  - **Port** – connection point for device
  - **Bus** - set of wires and rigidly defined communication protocol
    - daisy chain or shared direct access
  - **Controller** (host adapter) – electronics that operate port, bus, device
    - Sometimes integrated
    - Sometimes separate circuit board (host adapter)
    - Contains processor, microcode, private memory, bus controller, etc
PC Bus Structure

- **Monitor**
- **Processor**
- **Graphics Controller**
- **Bridge/Memory Controller**
- **Cache**
- **Memory**
- **IDE Disk Controller**
- **Expansion Bus Interface**
- **SCSI Controller**
- **Disk**
- **Keyboard**
- **Parallel Port**
- **Serial Port**

**Small Computer System Interface**

- **SCSI Bus**
- **Disk**
- **Expansion Bus**

**Complex Specialized Controller, Separate Circuit Board**

**Common PC System Bus**

**Bus for Slow Devices**

**Complex Specialized Controller, Separate Circuit Board**
I/O Controllers

- I/O instructions control devices
- Devices have registers where *device drivers* place commands, addresses, and data to write, or read data from registers after command execution
- Devices have addresses, used by
  - Direct I/O instructions
  - Memory-mapped I/O
    - Device data and command registers mapped to processor address space
    - Especially for large address spaces (graphics)
## Device I/O Locations

<table>
<thead>
<tr>
<th>I/O address range (hexadecimal)</th>
<th>device</th>
</tr>
</thead>
<tbody>
<tr>
<td>000–00F</td>
<td>DMA controller</td>
</tr>
<tr>
<td>020–021</td>
<td>interrupt controller</td>
</tr>
<tr>
<td>040–043</td>
<td>timer</td>
</tr>
<tr>
<td>200–20F</td>
<td>game controller</td>
</tr>
<tr>
<td>2F8–2FF</td>
<td>serial port (secondary)</td>
</tr>
<tr>
<td>320–32F</td>
<td>hard-disk controller</td>
</tr>
<tr>
<td>378–37F</td>
<td>parallel port</td>
</tr>
<tr>
<td>3D0–3DF</td>
<td>graphics controller</td>
</tr>
<tr>
<td>3F0–3F7</td>
<td>diskette-drive controller</td>
</tr>
<tr>
<td>3F8–3FF</td>
<td>serial port (primary)</td>
</tr>
</tbody>
</table>
Communication between Host and Controller

? How do I/O devices interact with host?

Polling vs. Interrupts
Polling

- For each byte of I/O
  - Host reads busy bit from status register until 0
  - Host sets read or write bit, and if write, copies data into data-out register
  - Host sets command-ready bit
  - Controller sets busy bit, executes transfer
  - Controller clears busy bit, error bit, command-ready bit when transfer done

- Step 1 is busy-wait cycle to wait for I/O from device
  - Reasonable if device is fast
  - But inefficient if device slow
  - CPU switches to other tasks?
    - But if miss a cycle data overwritten / lost

Problems with Polling?
Interrupts

- CPU **Interrupt-request line** triggered by I/O device
  - Checked by processor after each instruction
  - Modern OS: >100 interrupts per second (single user system)

- Defer interrupt handling during critical processes
- Efficient way to dispatch to proper interrupt handler
- Multilevel Interrupts: Priorities
Interrupts

• Interrupt handler receives interrupts
  • **Maskable** to ignore or delay some interrupts
• Interrupt vector to dispatch interrupt to correct handler
  • Context switch at start and end
  • Based on priority
  • Some **nonmaskable**
  • Interrupt chaining if more than one device at same interrupt number
# Intel Pentium Event-Vector Table

<table>
<thead>
<tr>
<th>vector number</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>divide error</td>
</tr>
<tr>
<td>1</td>
<td>debug exception</td>
</tr>
<tr>
<td>2</td>
<td>null interrupt</td>
</tr>
<tr>
<td>3</td>
<td>breakpoint</td>
</tr>
<tr>
<td>4</td>
<td>INTO-detected overflow</td>
</tr>
<tr>
<td>5</td>
<td>bound range exception</td>
</tr>
<tr>
<td>6</td>
<td>invalid opcode</td>
</tr>
<tr>
<td>7</td>
<td>device not available</td>
</tr>
<tr>
<td>8</td>
<td>double fault</td>
</tr>
<tr>
<td>9</td>
<td>coprocessor segment overrun (reserved)</td>
</tr>
<tr>
<td>10</td>
<td>invalid task state segment</td>
</tr>
<tr>
<td>11</td>
<td>segment not present</td>
</tr>
<tr>
<td>12</td>
<td>stack fault</td>
</tr>
<tr>
<td>13</td>
<td>general protection</td>
</tr>
<tr>
<td>14</td>
<td>page fault</td>
</tr>
<tr>
<td>15</td>
<td>(Intel reserved, do not use)</td>
</tr>
<tr>
<td>16</td>
<td>floating-point error</td>
</tr>
<tr>
<td>17</td>
<td>alignment check</td>
</tr>
<tr>
<td>18</td>
<td>machine check</td>
</tr>
<tr>
<td>19–31</td>
<td>(Intel reserved, do not use)</td>
</tr>
<tr>
<td>32–255</td>
<td>maskable interrupts</td>
</tr>
</tbody>
</table>
Direct Memory Access (DMA)

- Used to avoid one byte at a time for large data movement
- Requires DMA controller
- Bypasses CPU to transfer data directly between I/O device and memory
- OS writes DMA command block into memory
  - Source writes location of command block to DMA controller
  - Bus mastering of DMA controller – grabs bus from CPU
  - When done, interrupts to signal completion
DMA Transfer

1. Device driver is told to transfer disk data to buffer at address X
2. Device driver tells disk controller to transfer C bytes from disk to buffer at address X
3. Disk controller initiates DMA transfer
4. Disk controller sends each byte to DMA controller
5. DMA controller transfers bytes to buffer X, increasing memory address and decreasing C until C = 0
6. When C = 0, DMA interrupts CPU to signal transfer completion
Device Variations

- Character-stream or block
  - Bytes transferred one by one or as a block of bytes
- Sequential or random-access
  - Data transferred in fixed order or seek at any location
- Synchronous or asynchronous (or both)
  - Data transfer with predictable response time or irregular/unpredictable time
- Sharable or dedicated
  - May be shared by several processes or not
- Speed of operation
  - read-write, read only, or write only
# I/O Structure

<table>
<thead>
<tr>
<th>Software</th>
<th>Hardware</th>
</tr>
</thead>
<tbody>
<tr>
<td>kernel I/O subsystem</td>
<td>Hardware devices</td>
</tr>
<tr>
<td>kernel</td>
<td></td>
</tr>
<tr>
<td>SCSI device driver</td>
<td>SCSI devices</td>
</tr>
<tr>
<td>keyboard device driver</td>
<td>keyboard</td>
</tr>
<tr>
<td>mouse device driver</td>
<td>mouse</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>PCI bus device driver</td>
<td>PCI bus</td>
</tr>
<tr>
<td>floppy device driver</td>
<td>floppy-disk drives</td>
</tr>
<tr>
<td>ATAPI device driver</td>
<td>ATAPI devices (disks, tapes, drives)</td>
</tr>
</tbody>
</table>

### Diagram

Diagram showing the relationship between software and hardware components, illustrating how various device drivers interact with the system kernel and hardware devices.
### Characteristics

<table>
<thead>
<tr>
<th>aspect</th>
<th>variation</th>
<th>example</th>
</tr>
</thead>
<tbody>
<tr>
<td>data-transfer mode</td>
<td>character block</td>
<td>terminal disk</td>
</tr>
<tr>
<td>access method</td>
<td>sequential random</td>
<td>modem CD-ROM</td>
</tr>
<tr>
<td>transfer schedule</td>
<td>synchronous</td>
<td>tape keyboard</td>
</tr>
<tr>
<td></td>
<td>asynchronous</td>
<td></td>
</tr>
<tr>
<td>sharing</td>
<td>dedicated</td>
<td>tape keyboard</td>
</tr>
<tr>
<td></td>
<td>sharable</td>
<td></td>
</tr>
<tr>
<td>device speed</td>
<td>latency</td>
<td></td>
</tr>
<tr>
<td></td>
<td>seek time</td>
<td></td>
</tr>
<tr>
<td></td>
<td>transfer rate</td>
<td></td>
</tr>
<tr>
<td></td>
<td>delay between operations</td>
<td></td>
</tr>
<tr>
<td>I/O direction</td>
<td>read only</td>
<td>CD-ROM graphics controller</td>
</tr>
<tr>
<td></td>
<td>write only</td>
<td>disk</td>
</tr>
<tr>
<td></td>
<td>read–write</td>
<td></td>
</tr>
</tbody>
</table>
Block and Character Devices

• **Block devices** include disk drives
  • Commands include read, write, seek
  • Access via a file-system interface

• **Character devices** include keyboards, mice, serial ports
  • Commands include `get()`, `put()`
  • Libraries layered on top allow line editing
Network Devices

• Varying enough from block and character to have own interface

• Unix and Windows NT/9x/2000 include socket interface
  • Connect local socket to remote address
  • Separates network protocol from network operation
  • Includes select() functionality
Blocking and Nonblocking I/O

- **Blocking** - process suspended until I/O completed
  - Easy to use and understand
  - Insufficient for some needs

- **Nonblocking** - I/O call returns as much as available
  - User interface, data copy (buffered I/O)
  - Implemented via multi-threading
  - Returns quickly with count of bytes read or written
  - `select()` to find if data ready then `read()` or `write()` to transfer

- **Asynchronous** - process runs while I/O executes
  - Difficult to use
  - I/O subsystem signals process when I/O completed
Kernel I/O Subsystem

What services for kernel I/O system are needed?

• Scheduling
• Buffering - store data in memory while transferring between devices
• Caching
• Spooling
• Device Reservation
Scheduling and Spooling

• Scheduling
  • Per-device queue
  • Fairness

• Buffering - store data in memory while transferring between devices
  • To cope with device speed mismatch
  • To cope with device transfer size mismatch
### Device Status Table

<table>
<thead>
<tr>
<th>Device</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>keyboard</td>
<td>idle</td>
</tr>
<tr>
<td>laser printer</td>
<td>busy</td>
</tr>
<tr>
<td>mouse</td>
<td>idle</td>
</tr>
<tr>
<td>disk unit 1</td>
<td>idle</td>
</tr>
<tr>
<td>disk unit 2</td>
<td>busy</td>
</tr>
</tbody>
</table>

- **request for laser printer**
  - address: 38546
  - length: 1372

- **request for disk unit 2**
  - file: xxx
  - operation: read
  - address: 43046
  - length: 20000

- **request for disk unit 2**
  - file: yyy
  - operation: write
  - address: 03458
  - length: 500
Sun Enterprise Device-Transfer Rates

- System Bus
- HyperTransport (32-pair)
- PCI Express 2.0 (×32)
- Infiniband (QDR 12X)
- Serial ATA (SATA-300)
- gigabit ethernet
- SCSI bus
- FireWire
- hard disk
- modem
- mouse
- keyboard

Logarithmic scale for comparison of transfer rates.
Caching, Spooling, Device Reservation, and Error Handling

- **Caching** - faster device holding copy of data
  - Always just a copy
  - Key to performance

- **Spooling** - hold output for a device
  - If device can serve only one request at a time

- **Device reservation** - provides exclusive access to a device
  - Watch out for deadlock

- **Error Handling**
  - Retry a read or write, for example
  - Track error frequencies, stop using device with increasing frequency of retry-able errors
Summary

• I/O Hardware
• I/O Interface
• Kernel Subsystem