Why is GoI relevant for ICC?

Kazushige Terui

terui@kurims.kyoto-u.ac.jp

RIMS, Kyoto University
Plan

Interactive computation in complexity
Gol as abstract machine
Girard's conjecture
A logspace Gol algorithm for atomic MLL
An origin of interactive computation

Composition of two logspace Turing machines:

- **Sequential composition**

\[ M_1 \rightarrow M_2 \rightarrow \]

does not work (due to large intermediate values)

- One has to compose them interactively:

\[ M_1 \leftrightarrow M_2 \rightarrow \]
Oracle Turing machines

- Oracle TMs work on $k + 1$ tapes ($k$ work-tapes + 1 query-tape).

- An oracle TM is $(\Sigma, Q, \delta)$, where
  - $0, 1, b \in \Sigma$; $q_I, q_{F0}, q_{F1}, q_{Q}, q_{A0}, q_{A1} \in Q$
  - $\delta : Q \setminus \{q_{Q}, q_{F0}, q_{F1}\} \times \Sigma^{k+1} \rightarrow Q \times \Sigma^{k+1} \times \{l, c, r\}^{k+1}$

- Each word $w \in \{0, 1\}^*$ is identified with an oracle $O_w$ (partial function):

  $O_w : \mathbb{N} \rightarrow \{0, 1\}$

  $i \mapsto$ the $i$th bit of $w$ if $i \leq |w|$

  undefined otherwise
Oracle Turing machines

Given $O : \mathbb{N} \rightarrow \{0, 1\}$ and $n \in \mathbb{N}$, $M$ works as follows:

1. Initialize all tapes
2. Write down $n$ in binary on the query-tape
3. If state $\neq q_Q, q_{Fi}$, proceed as specified by $\delta$
4. If state $= q_Q$, then
   - let $i = O([\text{query-tape}])$ in state $:= q_{Ai}$
5. If state $= q_{Fi}$ ($i \in \{0, 1\}$), output $i$ and halt.
6. Goto 3
Properties of OTM

Def: $M$ is downward closed (d-closed) if for every $w \in \{0, 1\}^*$, $M(O_w, n)$ halts, $m \leq n \implies M(O_w, m)$ halts.

Def: $M$ is bounded if for every $w \in \{0, 1\}^*$, $\max\{n : M(O_w, n) \text{ halts}\}$ (the output length) exists.

Prop: Every bounded d-closed $M$ computes a function

$$F : \{0, 1\}^* \rightarrow \{0, 1\}^*$$

such that $M(O_w, n) = n\text{th bit of } F(w)$. 
Properties of OTM

**Def:** An OTM $M$ works in space $f : \mathbb{N} \rightarrow \mathbb{N}$ if for every $w \in \{0, 1\}^*$,

$M(O_w, n)$ halts $\implies$ \# (used cells) $\leq f(|w|)$.

**Fact:** If $M$ works in $f$, then

the output length $\leq 2^{f(n)}$

where $n$ is the input length. In particular, if $M$ works in $f(n) = k \log n$, then

the output length $\leq n^k$.

**Prop:** Logspace bounded d-closed OTMs compose.
TM\text{s vs Functional Programs}

- For TMs, there are two ways of composition:
  - \textbf{Sequential:} time-efficient
  - \textbf{Interactive:} space-efficient

- For functional programs, there are two ways of evaluation:
  - \textbf{Sequential (\(\beta\)-reduction):} time-efficient
  - \textbf{Interactive (token machines):} space-efficient

while there is only one \textit{canonical composition}:

\[ M_1 \circ M_2 = \lambda x. M_1(M_2 x). \]

- The latter might shed a new light on time-space trade-off.
GoI as Abstract Machine

- Interaction Abstract Machine (Danos, Regnier, …)

- An exponential signature is a binary tree with leaves labeled by \( d, 0, 1 \).

- A configuration is \((B, S)\), where
  - \(B\) is a stack made of exponential signatures
  - \(S\) is a stack made of exponential signatures, \(l\) and \(r\)

- Given a proof net, a run starts at a conclusion link \(s\) with initial configuration \((\epsilon, S')\). It is successful if it returns back to a conclusion link \(s'\) with \((\epsilon, S'')\) (notation: \((s, \epsilon, S') \rightarrow (s', \epsilon, S'')\)).

- Invariance: Suppose that an MELL proof \(\pi_0\) reduces to \(\pi_1\) by closed reduction. Then \((s, \epsilon, S') \rightarrow (s', \epsilon, S'')\) on \(\pi_0\) iff the same holds on \(\pi_1\).
Elementary (Multiplicative) Linear Logic

- EMLL = 2nd order MLL + monoidal functorial !:

\[
\begin{align*}
A_1, \ldots, A_n & \vdash B \\
!A_1, \ldots, !A_n & \vdash !B \\
!A, !A, \Gamma & \vdash B \\
!A, \Gamma & \vdash B \\
\Gamma & \vdash B
\end{align*}
\]

- EMLL corresponds to the elementary recursive functions (Girard 98, Mairson-Terui 03).

- GoI studied by (Baillot-Pedicini 00).

- In EMLL proof nets, pax (auxiliary doors of !-boxes) can be replaced with dereliction.
Elementary (Multiplicative) Linear Logic

Example:

\[
\begin{align*}
B & := \alpha \otimes \alpha \otimes \alpha \\
N_A & := !(A \otimes A) \otimes !(A \otimes A) \\
\text{true} & := \lambda x \otimes y. x \otimes y \quad : B \\
\text{false} & := \lambda x \otimes y. y \otimes x \quad : B \\
\text{neg} & := \lambda b \lambda x \otimes y. b(y \otimes x) \quad : B \rightarrow B \\
\text{even} & := \lambda n. n \otimes \text{neg true} \quad : N_B \rightarrow B 
\end{align*}
\]
GoI for MLL proof nets

- Girard’s conjecture: MLL proof nets are normalizable via (variant of) GoI in Logspace.

- Negative solution (Terui, Mairson 02): The following question is complete for $P$:

  Given two proof nets $\pi_1, \pi_2$, does $\pi_1 \equiv_\beta \pi_2$ hold?

  since boolean circuits are encodable in MLL.

\[
T \ := \ true \otimes false \quad F \ := \ false \otimes true \\
NEG \ := \ \lambda b \otimes \overline{b} \cdot \overline{b} \otimes b \\
CNTR \ := \ \lambda b \otimes \overline{b} \cdot (true \otimes false) \otimes \overline{b} \cdot (false \otimes true) \\
CONJ \ := \ \lambda b \otimes \overline{b} \cdot \lambda c \otimes \overline{c}. \\
\]

let $u \otimes v = b(c \otimes false)$, $\overline{u} \otimes \overline{v} = \overline{b}(true \otimes \overline{c})$ in 

$u \otimes (\overline{u} \circ v \circ \overline{v} \circ false)$
Atomic MLL

- **Theorem** (Mairson): Normalization in Atomic MLL (where all axioms are of atomic type) is complete for Logspace.

- For logspace computation, only a constant number of pointers are available, since each pointer is already of logarithmic size.

- Stacks are not available for tall proof nets.

- Mairson’s stack-free algorithm.
Conclusion

- Gol is implicit in composition of logspace TMs (will be further discussed in Ulrich’s talk).

- Gol leads to a space-efficient abstract machine.

- MLL is complete for P, whereas atomic MLL is complete for Logspace; stack-free Gol works for the latter.

- Work not mentioned:
  - (Dal Lago 05) uses context semantics for verification of time complexity of programs.
  - (Schöpp 06, 07) combine Gol with Hofmann realizability to show logspace completeness of subsystems of LFPL and Bounded Affine Logic.