Why is GoI relevant for ICC?

Kazushige Terui

terui@kurims.kyoto-u.ac.jp

RIMS, Kyoto University

Plan

- Interactive computation in complexity
- GoI as abstract machine
- Girard's conjecture
- A logspace GoI algorithm for atomic MLL

An origin of interactive computation

- Composition of two logspace Turing machines:
  - Sequential composition

$$\longrightarrow \boxed{M_1} \longrightarrow \boxed{M_2} \longrightarrow$$

  does not work (due to large intermediate values)
  - One has to compose them interatcively:

$$\longrightarrow \boxed{M_1} \rightleftarrows \boxed{M_2} \longrightarrow$$

Oracle Turing machies

- Oracle TMs work on $k+1$ tapes (k work-tapes + 1 query-tape).
- An oracle TM is (Σ, Q, δ), where
  - $0, 1, b \in \Sigma$; $q_I, q_{F0}, q_{F1}, q_Q, q_{A0}, q_{A1} \in Q$
  - $\delta : Q \backslash \{q_Q, q_{F0}, q_{F1}\} \times \Sigma^{k+1} \longrightarrow Q \times \Sigma^{k+1} \times \{l, c, r\}^{k+1}$
- Each word $w \in \{0,1\}^*$ is identified with an oracle O_w (partial function):

$$\begin{array}{rccl} O_w : & \mathbf{N} & \rightharpoonup & \{0,1\} \\ & i & \mapsto & \text{the } i\text{th bit of } w \text{ if } i \leq |w| \\ & & & \text{undefined otherwise} \end{array}$$

Oracle Turing machies

- Given $O : \mathbf{N} \rightharpoonup \{0,1\}$ and $n \in \mathbf{N}$, 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 δ
  4. If state $= q_Q$, then
     let $i = O(\lceil$query-tape$\rceil)$ 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, $m \le n \Longrightarrow M(O_w, m)$ halts.
- Def: M is bounded if for every $w \in \{0,1\}^*$, $max\{n : M(O_w, n)$ halts$\}$ (the output length) exists.
- Prop: Every bounded d-closed M computes a function

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

such that $M(O_w, n) = n$th bit of $F(w)$.

Properties of OTM

- Def: An OTM M works in space $f : \mathbf{N} \longrightarrow \mathbf{N}$ if for every $w \in \{0,1\}^*$,
  $M(O_w, n)$ halts $\Longrightarrow \sharp$ (used cells) $\leq f(|w|)$.
- Fact: If M works in f, then

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

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

  $$\text{the output length} \leq n^k.$$

- Prop: Logspace bounded d-closed OTMs compose.

TMs vs Functional Programs

- For TMs, there are two ways of composition:
  - Sequential: time-efficient
  - Interactive: space-efficient
- For functional programs, there are two ways of evaluation:
  - Sequential (β-reduction): time-efficient
  - Interactive (token machines): space-efficient

  while there is only one 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

- Intaraction 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 (ϵ, S). It is successful if it returns back to a conclusion link s' with (ϵ, S') (notation: $(s, \epsilon, S) \longrightarrow (s', \epsilon, S')$).
- Invariance: Suppose that an MELL proof π_0 reduces to π_1 by closed reduction. Then $(s, \epsilon, S) \longrightarrow (s', \epsilon, S')$ on π_0 iff the same holds on π_1.

Elementary (Multiplicative) Linear Logic

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

$$\frac{A_1, \ldots, A_n \vdash B}{!A_1, \ldots, !A_n \vdash !B} \qquad \frac{!A, !A, \Gamma \vdash B}{!A, \Gamma \vdash B} \qquad \frac{\Gamma \vdash B}{!A, \Gamma \vdash B}$$

- 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:

$$
\begin{array}{rcll}
\mathbf{B} & := & \alpha \otimes \alpha \multimap \alpha \otimes \alpha & \\
\mathbf{N}_A & := & !(A \multimap A) \multimap !(A \multimap A) & \\
\mathsf{true} & := & \lambda x \otimes y . x \otimes y & : \mathbf{B} \\
\mathsf{false} & := & \lambda x \otimes y . y \otimes x & : \mathbf{B} \\
neg & := & \lambda b \lambda x \otimes y . b(y \otimes x) & : \mathbf{B} \multimap \mathbf{B} \\
even & := & \lambda n . n \, !neg \, \mathsf{true} & : \mathbf{N}_{\mathbf{B}} \multimap \mathbf{B}
\end{array}
$$

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 π_1, π_2, does $\pi_1 =_\beta \pi_2$ hold?

  since boolean circuits are encodable in MLL.

$$
\begin{aligned}
T &:= \mathsf{true} \otimes \mathsf{false} \qquad F := \mathsf{false} \otimes \mathsf{true} \\
NEG &:= \lambda b \otimes \overline{b}.\overline{b} \otimes b \\
CNTR &:= \lambda b \otimes \overline{b}.b(\mathsf{true} \otimes \mathsf{false}) \otimes \overline{b}(\mathsf{false} \otimes \mathsf{true}) \\
CONJ &:= \lambda b \otimes \overline{b}.\lambda c \otimes \overline{c}. \\
&\quad\; let\ u \otimes v = b(c \otimes \mathsf{false}),\ \overline{u} \otimes \overline{v} = \overline{b}(\mathsf{true} \otimes \overline{c})\ in \\
&\quad\; u \otimes (\overline{u} \circ v \circ \overline{v} \circ \mathsf{false})
\end{aligned}
$$

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 logarithmetic size.
- Stacks are not available for tall proof nets.
- Mairson's stack-free algorithm.

Conclusion

- GoI is implicit in composition of logspace TMs (will be further discussed in Ulrich's talk).
- GoI leads to a space-efficient abstract machine.
- MLL is complete for P, whereas atomic MLL is complete for Logspace; stack-free GoI works for the latter.
- Work not mentioned:
  - (Dal Lago 05) uses context semantics for verification of time complexity of programs.
  - (Schöpp 06, 07) combine GoI with Hofmann realizability to show logspace completeness of subsystems of LFPL and Bounded Affine Logic.