Implicit Complexity: to infinity... and beyond!

Complexité Implicite : vers l'infini et au-delà !

Romain Péchoux,
IPT Mocqua - CNRS, Inria, Université de Lorraine - LORIA

Séminaire PPS

February 4th, 2021

Today's talk

On my work in Inria team MOCQUA (Classical and QUAntum MOdels of Computation):

- Study of computational models
- and their properties (including complexity)

Today, we will focus on:

1. a brief overview of Implicit Computational Complexity
2. a characterization of type-2 polynomial time

Computational Complexity (CC)

Computational Complexity (CC) studies problems/functions wrt resource usage.

The Universe of mathematical functions

(Images: NASA)

Computational Complexity (CC)

Computational Complexity (CC) studies problems/functions wrt resource usage.

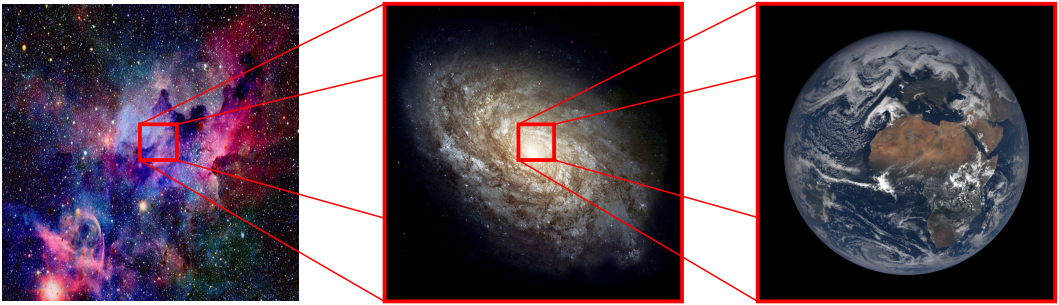
The Universe of mathematical functions

The Galaxy of **computable** functions

(Images: NASA)

Computational Complexity (CC)

Computational Complexity (CC) studies problems/functions wrt resource usage.



The Universe of mathematical functions

The Galaxy of **computable** functions

The Planet of **tractable** functions

Assume Cobham-Edmonds thesis: **tractable/feasible** = **polynomial time**.

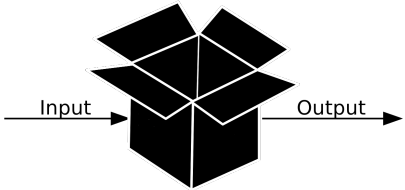
(Images: NASA)

Implicit computational complexity (ICC)

ICC: Subfield of CC aiming at providing characterizations of complexity classes:

- **machine-independent**
- with **no prior knowledge** on the complexity analyzed codes

If the characterization is **tractable** then ICC provides **automatic** static complexity analysis methods for **high level** PL.



State of the art:

- 30 years of intensive research,
- hundreds of publications,
- some academic tools
  - (Costa, SPEED, TcT, ...).

The ICC approach

ICC criterion

Take your favourite PL $\mathcal{L}$ and your favourite complexity class $\mathcal{C}$:

$$\mathcal{R} \subseteq \mathcal{L} \text{ is an } \textbf{ICC criterion} \text{ if } \{[\![\mathrm{p}]\!] \mid \mathrm{p} \in \mathcal{R}\} = \mathcal{C}.$$

Examples of complexity class $\mathcal{C}$

- P, FP,
- PSPACE, FPSPACE,
- EXP, 2-EXP, ..., ELEMENTARY,
- NP,
- NC^0, NC^1, ..., NC
- PP, BPP, EQP, BQP, ...

Examples of programming language $\mathcal{L}$

- lambda-calculi,
- term rewrite systems,
- process calculi,
- reactive programs,
- imperative and OO programs,
- probabilistic and quantum programs.

A bunch of techniques (1/2)

Some ICC criteria

- **function algebra**: [Cobham65], [Bellantoni-Cook92], [Clote99] for a survey
- **linear logic** based approaches
  - light logics: LLL [Girard87], LAL [Asperti98], DLAL [Baillot-Terui04],
  - soft logics: SLL [Lafont04], STA [Gaboardi-Ronchi Della Rocca07],
  - non size-increasing [Hofmann99].
- **"potential"** based methods
  - interpretations: "quasi" [Bonfante-Marion-Moyen11], "sup" [Marion-Péchoux09], higher-order [Baillot-Dal Lago16],
  - amortized resource analysis: [Jost et al.10], [Hoffmann-Hofmann10],
  - sized-types: [Vasconcelos08], [Avanzini-Dal Lago17],
  - cost semantics: [Danner et al.15].

A bunch of techniques (2/2)

Some ICC criteria

- **control flow (tiering-based)** techniques:
  - safe recursion [Bellantoni-Cook92],
  - ramified reccurence [Leivant-Marion94],
  - tiering [Marion11],
  - read-only/write-only: [Jones01], [De Carvalho-Simonsen14].
- **matrix-based** type systems:
  - μ-measure [Niggl-Wunderlich06],
  - mwp bounds [Kristiansen-Jones09], resource control graphs [Moyen09].
- **empirical approaches** (some of them using abstract interpretations): COSTA [Albert et al.07], SPEED [Gulwani09], TcT[Avanzini-Moser-Schaper16].

Main techniques (1/2): typing

Tractable functions

Characterized by all techniques by preventing exponentiation,
i.e. by **preventing the iteration** of methods duplicating the size of their inputs.

- Prevent iteration with **a type discipline**:
  - $!A \multimap \S A$ in LAL,
  - $\mathbf{1} \to \mathbf{0}$ in tier-based approaches,
  - Read-Only $\to$ Write-Only in Jones/Simonsen
  - $\begin{bmatrix} \ddots & & \\ & P & \\ & & \ddots \end{bmatrix}$ in mwp (whereas $\begin{bmatrix} \ddots & & \\ & M & \\ & & \ddots \end{bmatrix}$ is required for iterability).

Main techniques (2/2): potentials

Tractable functions

Characterized by all techniques by preventing exponentiation,
i.e. by **preventing the iteration** of methods duplicating the size of their inputs.

- By using **a potential-based constraints** implying a decrease along reduction:

$$\begin{array}{rcccccccc} & t_1 & \to & t_2 & \to & \dots & \to & t_n \\ \mathsf{P} \geq & [t_1] & \geq & [t_2] & \geq & \dots & \geq & [t_n] \end{array}$$

  - (polynomial) interpretations-based methods,
  - amortized resource analysis,
  - ert-transformers method [Kaminski et al.06],
  - sized-types.

Intensional limits

Definition [Intensional completeness]

A characterization is intensionally complete if any tractable algorithm computing this function is accepted.

Theorem [Hajek79]

Providing an intensionally-complete characterization of tractable functions is a Σ_0^2-complete problem.

However, for automation purpose, the studied characterizations are decidable (even better tractable).

Observation

Hence there are false negative.

Beyond ICC: extensions

Intensional improvements

- Soft Type Assignment [Gaboardi-Ronchi Della Rocca07]
- Dual Light Affine Logic [Baillot-Terui04]
- Sup-interpretations [Marion-Péchoux09]

Adaptations of existing tools

- Tiering on imperative programs [Marion11], [Marion-Leivant13]
- Tiering on OO programs [Hainry-Péchoux18]
- Interpretations of HO-TRS (STTRS) [Baillot-Dal Lago12]

Extensions to new paradigms

- Concurrent systems
  - Light logics and multi-threads [Amadio-Madet11]
  - Soft logics and processes [Martini-Dal Lago-Sangiorgi16]
- Probabilistic programs: [Avanzini-Dal Lago-Ghyselen19]
- Quantum programs [Dal Lago-Masini-Zorzi10]
- Real functions [Bournez-Gomaa-Hainry11]
- Coinductive data [Gaboardi-Péchoux15]

Summary on ICC

Strong links with other research domains:

- Termination techniques (often coming from and/or combined with)
- Computability theory (Primrec, undecidable classes, ...)
- Finite model theory (common goals)
- Static analysis (type systems, abstract interpretations, empirical approaches)

A **survey on ICC** in my HDR, available at `https://members.loria.fr/RPechoux/`

ICC: the case of "infinite" data (1/2)

Several **research directions**:

- streams, infinite structures,
- coinductive data types,
- higher order functions,
- real numbers and their representations,
- higher order complexity.

ICC: the case of "infinite" data (2/2)

Some corresponding **results**:

- Extension of interpretations to stream data [Gaboardi-Péchoux09]
- Extension of LALC to codinductive data [Gaboardi-Péchoux15]
- Characterization of P/poly in the infinitary lambda calculus [Mazza14]
- The parsimonious lambda-calculus[Mazza-Terui15]:
  - Characterization of P/poly [Mazza-Terui15]
  - Characterization of P($\mathbb{R}$) [Hainry-Mazza-Péchoux20]
- Characterization of BFF [Férée-Hainry-Hoyrup-Péchoux15]
  - first PL based characterization of type-2 polytime
  - but not tractable (type-2 polynomial inequalities).

A reminder on type-2 polynomial time

BFF was introduced by Melhorn in 1976.

Theorem [Cook and Urquhart [1989]]

$\mathtt{BFF} = \lambda(\mathtt{FP} \cup \{\mathcal{R}\})_2$

$\mathcal{R}$ is a type-2 bounded iterator:

$$\mathcal{R}(\epsilon, a) = a$$
$$\mathcal{R}(ix, a) = \min(\phi(ix, \mathcal{R}(x, a)), \psi(ix))$$

Theorem [Cook and Kapron [1990]]

The set of functionals computable by an OTM in time $P(|\phi|, |\mathbf{a}|)$ is exactly BFF.

2nd order polynomials and size function are defined by:

- $P(X_1, X_0) ::= c \in \mathbb{N} \mid X_0 \mid X_1(P) \mid P + P \mid P \times P$
- $|\phi|(n) = \max_{|x| \leq n} |\phi(x)|$

How to get rid of type-2 polynomials?

Definition [Oracle Polynomial Time (OPT) [Cook92]]

Let $n^{\phi,\mathbf{a}}$ be the biggest size of $\mathbf{a}$ and of an oracle's answer in the run of $M(\phi, \mathbf{a})$.
An OTM is in OPT if its runtime is bounded by $P(n^{\phi,\mathbf{a}})$, for some type-1 polynomial P.

BFF $\subsetneq$ OPT as it contains exponential functions.

Theorem [Kapron and Steinberg [2018]]

$\mathtt{BFF} = \lambda(\mathrm{OPT} \cap \mathrm{FLR})_2 = \lambda(\mathrm{OPT} \cap \mathrm{FLAR})_2$

- FLR = Finite Length Revision
- FLAR = Finite LookAhead Revision

Finite Length Revision

Definition [Finite Length Revision - Kawamura and Steinberg [2017]]

An OTM is in FLR, if, for any input, the number of times the oracle answer is bigger than all of the previous oracle answers is bounded by a constant.

Example

```
while (x>0){
        y = φ(x);
        x = x-1;
}
```

not (FLR) if $\phi \searrow$

Example

```
while (x<n && y<8){
        y = φ(x);
        x = x+1;
}
```

(FLR) with constant 8

Finite LookAhead Revision

Definition [Finite LookAhead Revision - Kapron and Steinberg [2018]]

An OTM is in FLAR, if, for any input, the number of times a query is posed whose size exceeds the size of all previous queries is bounded by a constant.

Example

```
while (x>0){
        y = φ(x);
        x = x-1;
}
```

(FLAR) with constant 0

Example

```
while (x<n && y<8){
        y = φ(x);
        x = x+1;
}
```

not (FLAR) for $\phi = \lambda z.4$

A tier-based characterization of BFF

Imperative PL with oracles and tier-based type system

(Expressions) $\mathtt{e} ::= \mathtt{x} \mid \mathtt{true} \mid \mathtt{false} \mid \mathtt{op}(\overline{\mathtt{e}}) \mid \phi(\mathtt{e} \upharpoonright \mathtt{e})$

(Statements) $\mathtt{st} ::= [\mathtt{x}{:=}\mathtt{e}]; \mid \mathtt{st}\ \mathtt{st} \mid \mathtt{while}(\mathtt{e})\{\mathtt{st}\} \mid \mathtt{if}(\mathtt{e})\{\mathtt{st}\}\mathtt{else}\{\mathtt{st}\}$

Let $[\![\mathsf{ST}]\!]$ be the set of functions computed by programs in ST (Safe and Terminating).

Theorem [Hainry-Kapron-Marion-Péchoux20]

$\mathtt{BFF} = \lambda([\![\mathsf{ST}]\!])_2$

Drawbacks

- lambda closure (for completeness)
- termination assumption (for soundness)

A new tier-based characterization of BFF

New tier-based PL

e^i: e is a type-i object.

$$
\begin{array}{lrcl}
\text{(Expressions)} & \mathtt{e} & ::= & \mathtt{x}^0 \mid \mathtt{op}(\overline{\mathtt{e}}) \mid \mathtt{x}^1(\mathtt{e} \upharpoonright \mathtt{e}) \\
\text{(Statements)} & \mathtt{st} & ::= & [\mathtt{x}^0 := \mathtt{e}]; \mid \mathtt{st\ st} \mid \mathtt{if}(\mathtt{e})\{\mathtt{st}\}\{\mathtt{st}\} \mid \mathtt{while}(\mathtt{e})\{\mathtt{st}\} \\
\text{(Procedures)} & \mathtt{P} & ::= & \mathtt{P}(\overline{\mathtt{x}^1}, \overline{\mathtt{x}^0})\{\mathtt{st\ \ return\ x}^0\} \\
\text{(Terms)} & \mathtt{t} & ::= & \mathtt{x}^i \mid \lambda \mathtt{x}^i.\mathtt{t} \mid \mathtt{t}_1 @ \mathtt{t}_2 \mid \mathtt{call\ P}(\overline{\{\mathtt{x}^0 \to \mathtt{t}^0\}}, \overline{\mathtt{t}^0}) \\
\text{(Programs)} & \mathtt{prog} & ::= & \mathtt{t}^0 \mid \mathtt{declare\ P\ in\ prog}
\end{array}
$$

- SAFE is the set of typable programs.
- SAFE_0 is the set of typable programs without lambda-abstraction.
- SN is the set of strongly normalizing programs.

Example

```
prog(X,x)= declare KS(Y, Z, v) {
                        w  :=  Y(ε ⊾ ε) ;
                        z  :=  ε ;
                        while (v != ε) {
                            v  :=  pred(v) ;
                            z  :=  Z(z ⊾ w)
                        }
                        return z
                    }
            in call KS({y → X @ y}, {y → X @ (X @ y)}, x)
```

- $[\![\mathrm{prog}]\!] \in (\mathbb{W} \to \mathbb{W}) \to \mathbb{W} \to \mathbb{W}$
- $[\![\mathrm{prog}]\!](f^{\mathbb{W}\to\mathbb{W}}, w^{\mathbb{W}}) = F_{|w|}(f)$ with $\begin{cases} F_0(f) = \epsilon \\ F_{n+1}(f) = (f \circ f)(F_n(f)^{\leq |f(1)|}) \end{cases}$
- $\mathrm{prog} \in \mathrm{SAFE}_0 \cap \mathrm{SN}$

A complete type-1 polynomial time characterization of BFF

Soundness and completeness without lambda closure:

Theorem [New! [Hainry, Kapron, Marion, Péchoux]]

$\mathtt{BFF} = [\![\mathsf{SAFE}_0 \cap \mathsf{SN}]\!] = [\![\mathsf{SAFE} \cap \mathsf{SN}]\!]$

Preserved for a Ptime instance of Size Change Termination [Lee-BenAmram-Jones01]:

Theorem [New! [Hainry, Kapron, Marion, Péchoux21]]

$\mathtt{BFF} = [\![\mathsf{SAFE}_0 \cap \mathsf{SCP}_S]\!] = [\![\mathsf{SAFE} \cap \mathsf{SCP}_S]\!]$

SCP_S can be decided in time quadratic in $|\mathtt{prog}|$ (using [BenAmram-Lee07]).

Theorem [New! [Hainry, Kapron, Marion, Péchoux21]]

- $\mathtt{prog} \in \mathsf{SAFE} \cap \mathsf{SCP}_S$ is Ptime-complete (using [Mairson04])
- $\mathtt{prog} \in \mathsf{SAFE}_0 \cap \mathsf{SCP}_S$ is in time cubic in $|\mathtt{prog}|$ (using [Hainry et al.20])

Conclusion

There are plenty of ICC-related open issues:

- Continue to extend techniques using
  - Simulation techniques/Improvement theory [Sands95],
  - Non idempotent intersection types [Kesner-Vial20],
  - Shape-analysis [Sagiv et al.02]
- Continue the development of practical tools
  - `ComplexityParser` [Hainry-Jeandel-Péchoux-Zeyen20]
- Develop extensions to Quantum programs
  - What kind of programs? [Péchoux-Perdrix-Rennela-Zamdzhiev20]
  - Existing ICC Probabilistic approaches [Avanzini-Dal Lago-Ghyselen19], [Avanzini-Moser-Schaper20]
  - Existing ICC Quantum approaches [Dal Lago-Masini-Zorzi10], [Yamakami18]
  - What is the role of entanglement?

Thank you for your attention !

Type system: simply-typed part

$$\frac{\Gamma_{\mathtt{W}}, \Omega, \Delta \vdash \overline{\mathtt{X}} : \overline{\mathtt{W} \to \mathtt{W}} \qquad \Gamma_{\mathtt{W}}, \Omega, \Delta \vdash \overline{\mathtt{x}}, \overline{\mathtt{y}}, \mathtt{x} : \overline{\mathtt{W}}}{\Gamma_{\mathtt{W}}, \Omega, \Delta \vdash \mathtt{P}(\overline{\mathtt{X}}, \overline{\mathtt{x}})\{\mathtt{st}\ \ \mathtt{return}\ \mathtt{x}\} : (\overline{\mathtt{W} \to \mathtt{W}}) \to \overline{\mathtt{W}} \to \mathtt{W}} \ (\mathrm{PR})$$

$$\frac{\Gamma_{\mathtt{W}}, \Omega, \Delta \vdash \mathtt{P}(\overline{\mathtt{X}}, \overline{\mathtt{x}})\{\ldots\} : (\overline{\mathtt{W} \to \mathtt{W}}) \to \overline{\mathtt{W}} \to \mathtt{W} \qquad \Gamma_{\mathtt{W}}, \Omega, \Delta \vdash \overline{\mathtt{c}} : \overline{\mathtt{W} \to \mathtt{W}} \qquad \Gamma_{\mathtt{W}}, \Omega, \Delta \vdash \overline{\mathtt{t}} : \overline{\mathtt{W}}}{\Gamma_{\mathtt{W}}, \Omega, \Delta \vdash \mathtt{call}\ \mathtt{P}(\overline{\mathtt{c}}, \overline{\mathtt{t}}) : \mathtt{W}} \ (\mathrm{CC})$$

$$\frac{\Gamma_{\mathtt{W}}(\mathtt{x}) = \mathtt{T}}{\Gamma_{\mathtt{W}}, \Omega, \Delta \vdash \mathtt{x} : \mathtt{T}} \ (\mathrm{VAR}) \qquad \frac{\Gamma_{\mathtt{W}}, \Omega, \Delta \vdash \mathtt{t}_1 : \mathtt{T} \to \mathtt{T}' \qquad \Gamma_{\mathtt{W}}, \Omega, \Delta \vdash \mathtt{t}_2 : \mathtt{T}}{\Gamma_{\mathtt{W}}, \Omega, \Delta \vdash \mathtt{t}_1 @ \mathtt{t}_2 : \mathtt{T}'} \ (\mathrm{APP})$$

$$\frac{\Gamma_{\mathtt{W}} \uplus \{\mathtt{x} : \mathtt{T}\}, \Omega, \Delta \vdash \mathtt{t} : \mathtt{T}'}{\Gamma_{\mathtt{W}}, \Omega, \Delta \vdash \lambda \mathtt{x}.\mathtt{t} : \mathtt{T} \to \mathtt{T}'} \ (\mathrm{ABS}) \qquad \frac{\Gamma_{\mathtt{W}} \uplus \{\mathtt{x} : \mathtt{W}\}, \Omega, \Delta \vdash \mathtt{t} : \mathtt{W}}{\Gamma_{\mathtt{W}}, \Omega, \Delta \vdash \{\mathtt{x} \mapsto \mathtt{t}\} : \mathtt{W} \to \mathtt{W}} \ (\mathrm{CLOS})$$

$$\frac{\Gamma_{\mathtt{W}}, \Omega, \Delta \vdash \mathtt{prog} : \mathtt{T} \qquad \Gamma, \Delta \vdash \mathrm{body}(\mathtt{p}) : (\mathbf{k}, \mathbf{k}_{in}, \mathbf{k}_{out}) \qquad \Omega(\mathtt{p}) = \langle \Gamma, (\mathbf{k}, \mathbf{k}_{in}, \mathbf{k}_{out}) \rangle}{\Gamma_{\mathtt{W}}, \Omega, \Delta \vdash \mathtt{declare}\ \mathtt{p}\ \mathtt{in}\ \mathtt{prog} : \mathtt{T}} \ (\mathrm{DEC})$$

Type system: tier-based part 1/2

$$\frac{\Gamma(\mathtt{x}) = \mathbf{k}}{\Gamma, \Delta \vdash \mathtt{x} : (\mathbf{k}, \mathbf{k}_{in}, \mathbf{k}_{out})} \text{ (VAR)}$$

$$\frac{\mathbf{k}_1 \to \cdots \to \mathbf{k}_{|\overline{\mathtt{e}}|} \to \mathbf{k} \in \Delta(\mathtt{op})(\mathbf{k}_{in}) \qquad \forall i \leq |\overline{\mathtt{e}}|,\ \Gamma, \Delta \vdash \mathtt{e}_i : (\mathbf{k}_i, \mathbf{k}_{in}, \mathbf{k}_{out})}{\Gamma, \Delta \vdash \mathtt{op}(\overline{\mathtt{e}}) : (\mathbf{k}, \mathbf{k}_{in}, \mathbf{k}_{out})} \text{ (OP)}$$

$$\frac{\Gamma, \Delta \vdash \mathtt{e}_1 : (\mathbf{k}, \mathbf{k}_{in}, \mathbf{k}_{out}) \qquad \Gamma, \Delta \vdash \mathtt{e}_2 : (\mathbf{k}_{out}, \mathbf{k}_{in}, \mathbf{k}_{out}) \qquad \mathbf{k} \prec \mathbf{k}_{in} \wedge \mathbf{k} \preceq \mathbf{k}_{out}}{\Gamma, \Delta \vdash \mathtt{X}(\mathtt{e}_1 \upharpoonright \mathtt{e}_2) : (\mathbf{k}, \mathbf{k}_{in}, \mathbf{k}_{out})} \text{ (OR)}$$

$$\frac{\Gamma, \Delta \vdash \mathtt{st} \ : (\mathbf{k}, \mathbf{k}_{in}, \mathbf{k}_{out})}{\Gamma, \Delta \vdash \mathtt{st} \ : (\mathbf{k}+\mathbf{1}, \mathbf{k}_{in}, \mathbf{k}_{out})} \text{ (SUB)}$$

$$\frac{\Gamma, \Delta \vdash \mathtt{st}_1 : (\mathbf{k}, \mathbf{k}_{in}, \mathbf{k}_{out}) \qquad \Gamma, \Delta \vdash \mathtt{st}_2 : (\mathbf{k}, \mathbf{k}_{in}, \mathbf{k}_{out})}{\Gamma, \Delta \vdash \mathtt{st}_1 \ ; \ \mathtt{st}_2 \ : (\mathbf{k}, \mathbf{k}_{in}, \mathbf{k}_{out})} \text{ (SEQ)}$$

Type system: tier-based part 2/2

$$\frac{\Gamma, \Delta \vdash \mathtt{x} : (\mathbf{k}_1, \mathbf{k}_{in}, \mathbf{k}_{out}) \qquad \Gamma, \Delta \vdash \mathtt{e} : (\mathbf{k}_2, \mathbf{k}_{in}, \mathbf{k}_{out}) \qquad \mathbf{k}_1 \preceq \mathbf{k}_2}{\Gamma, \Delta \vdash \mathtt{x} := \mathtt{e} \ : (\mathbf{k}_1, \mathbf{k}_{in}, \mathbf{k}_{out})} \ (\mathrm{ASG})$$

$$\frac{\Gamma, \Delta \vdash \mathtt{e} : (\mathbf{k}, \mathbf{k}_{in}, \mathbf{k}_{out}) \qquad \forall w \in \{\mathtt{true}, \mathtt{false}\},\ \Gamma, \Delta \vdash \mathtt{st}_w : (\mathbf{k}, \mathbf{k}_{in}, \mathbf{k}_{out})}{\Gamma, \Delta \vdash \mathtt{if}(\mathtt{e})\{\mathtt{st}_{\mathtt{true}}\}\ \mathtt{else}\ \{\mathtt{st}_{\mathtt{false}}\} \ : (\mathbf{k}, \mathbf{k}_{in}, \mathbf{k}_{out})} \ (\mathrm{CND})$$

$$\frac{\Gamma, \Delta \vdash \mathtt{e} : (\mathbf{k}, \mathbf{k}_{in}, \mathbf{k}) \qquad \Gamma, \Delta \vdash \mathtt{st} : (\mathbf{k}, \mathbf{k}, \mathbf{k}) \qquad \mathbf{1} \preceq \mathbf{k}}{\Gamma, \Delta \vdash \mathtt{while}(\mathtt{e})\{\mathtt{st}\} \ : (\mathbf{k}, \mathbf{k}_{in}, \mathbf{0})} \ (\mathrm{WINIT})$$

$$\frac{\Gamma, \Delta \vdash \mathtt{e} : (\mathbf{k}, \mathbf{k}_{in}, \mathbf{k}_{out}) \qquad \Gamma, \Delta \vdash \mathtt{st} : (\mathbf{k}, \mathbf{k}, \mathbf{k}_{out}) \qquad \mathbf{1} \preceq \mathbf{k} \preceq \mathbf{k}_{out}}{\Gamma, \Delta \vdash \mathtt{while}(\mathtt{e})\{\mathtt{st}\} \ : (\mathbf{k}, \mathbf{k}_{in}, \mathbf{k}_{out})} \ (\mathrm{WH})$$