Replication, Recursion and Concurrency

J. Garrett Morris

University of Edinburgh

Linear logic provides a foundation for session typing. Well-typed programs are:

- Deadlock-free (corresponding to cut elimination)
- Race-free (derived from linearity)

Development includes π-DILL[1] and CP[2].

[1] Caires and Pfenning 2010, and following
[2] Wadler 2012

Servers and linearity

Linear logic exponentials provide π-calculus like process replication

- !-types correspond to replicated processes
- ?-types correspond to uses of replicated processes

Safe to be shared, as processes always identical.

Servers and linearity

Linear logic exponentials provide π-calculus like process replication

- !-types correspond to replicated processes
- ?-types correspond to uses of replicated processes

Safe to be shared, as processes always identical.

But, expressiveness is limited:

- Unable to express recursive sessions
- Unable to express stateful servers

$$\nu x.(!x(y).P \mid a[b].(?x[y].Q \mid ?x[z].R) \quad \Rightarrow$$

Note: two uses of x on right-hand side.

$$\nu x.(!x(y).P \mid a[b].(?x[y].Q \mid ?x[z].R) \quad \Rightarrow$$
$$\nu x.(!x(y).P \mid \nu x'.(!x'(y).P \mid a[b].(?x[y].Q \mid ?x'[z].R)) \quad \Rightarrow$$

Note: two uses of x on right-hand side.
Reduction duplicates server process as necessary . . .

$$\begin{array}{l}
\nu x.(!x(y).P \mid a[b].(?x[y].Q \mid ?x[z].R) \quad \Rightarrow \\
\nu x.(!x(y).P \mid \nu x'.(!x'(y).P \mid a[b].(?x[y].Q \mid ?x'[z].R)) \quad \Rightarrow \\
a[b].(\nu y.(P \mid Q) \mid \nu z.(P\{z/y\} \mid R))
\end{array}$$

Note: two uses of x on right-hand side.

Reduction duplicates server process as necessary . . .

. . . and then reduces as expected.

Adding recursion to sequent calculi well-studied (particularly w.r.t. modal logics[3], but recently in linear logic[4]).

- Sufficient to express recursive sessions
- Provides more expressive servers than does replication

[3] Bradfield and Sterling 2007

[4] Baelde and Miller 2007, Baelde 2012

Adding recursion to sequent calculi well-studied (particularly w.r.t. modal logics[3], but recently in linear logic[4]).

- Sufficient to express recursive sessions
- Provides more expressive servers than does replication

Preserves properties from linear logic:

- Deadlock-free (still enjoys cut elimination)
- Race-free (derived from linearity)

[3] Bradfield and Sterling 2007
[4] Baelde and Miller 2007, Baelde 2012

Adding recursion to linear logic

Two new, dual proposition types (X free in B)

- Least fixed points $\mu X.B$
- Greatest fixed points $\nu X.B$

with $(\nu X.B)^{\perp} = \mu X.(B^{\perp})$

Adding recursion to linear logic

Two new, dual proposition types (X free in B)

- Least fixed points $\mu X.B$
- Greatest fixed points $\nu X.B$

with $(\nu X.B)^\perp = \mu X.(B^\perp)$

Interpretation (inexact):

- ν-types correspond to *unbounded* looping
- μ-types correspond to *bounded* looping

Recursion and replication

Recursion provides replication. A value $?A$ can be:

- *Derelicted*, from $?A$ to A;
- *Weakened*, from $?A$ to $\bot$; and,
- *Contracted*, from $?A$ to $?A \parr ?A$

Contraction is implicit in CP syntax, but essential to CP semantics.

Recursion provides replication. A value $?A$ can be:

- *Derelicted*, from $?A$ to A;
- *Weakened*, from $?A$ to $\bot$; and,
- *Contracted*, from $?A$ to $?A \parr ?A$

Contraction is implicit in CP syntax, but essential to CP semantics.

We can capture these in a recursive type:

$$(\!|?A|\!) = \mu X.\bot \oplus A \oplus (X \parr X)$$

?-types can be coded recursively:

$$\llparenthesis ?A \rrparenthesis = \mu X . \bot \oplus A \oplus (X \parr X)$$

and !-types are dual to ?-types:

$$(?A)^{\bot} = !(A^{\bot})$$

?-types can be coded recursively:

$$(\!|?A|\!) = \mu X.\bot \oplus A \oplus (X \parr X)$$

and !-types are dual to ?-types:

$$(?A)^\bot = !(A^\bot)$$

So, we get

$$\begin{aligned}(\!|!A|\!) &= (\!|?(A^\bot)|\!)^\bot \\ &= (\mu X.\bot \oplus A^\bot \oplus (X \parr X))^\bot \\ &= \nu X.1 \,\&\, A \,\&\, (X \otimes X)\end{aligned}$$

Recursion and replication

$$(\!|?A|\!) = \mu X.\bot \oplus A \oplus (X \otimes X)$$
$$(\!|!A|\!) = \nu X.1 \,\&\, A \,\&\, (X \otimes X)$$

- Every proof of $!A$ can be transformed to a proof of $(\!|!A|\!)$
- And every proof of $?A$ can be transformed to a proof of $(\!|?A|\!)$ (making contraction and weakening explicit)

Recursion and replication

$$(\!|?A|\!) = \mu X.\bot \oplus A \oplus (X \otimes X)$$
$$(\!|!A|\!) = \nu X.1 \,\&\, A \,\&\, (X \otimes X)$$

- Every proof of $!A$ can be transformed to a proof of $(\!|!A|\!)$
- And every proof of $?A$ can be transformed to a proof of $(\!|?A|\!)$ (making contraction and weakening explicit)
- But not every proof of $(\!|!A|\!)$ can be tranformed to a proof of $!A$ (two processes resulting from contraction need not be identical)

Example: A counting server

Goal is to define a server P that provides an increasing series of naturals.

$$\nu s.(P \mid a[b].(b[c].(?s[x].c \leftrightarrow x \mid ?s[y].b \leftrightarrow y) \mid ?s[z].a \leftrightarrow z))$$

(Transform clients $?s[x].Q$ depending on encoding of servers)
Should be able to produce any ordering of 1,2,3 on a.

Example: A counting server

Goal is to define a server P that provides an increasing series of naturals.

$$\nu s.(P \mid a[b].(b[c].(?s[x].c \leftrightarrow x \mid ?s[y].b \leftrightarrow y) \mid ?s[z].a \leftrightarrow z))$$

Should be able to produce any ordering of 1,2,3 on a.

Replication insufficient

- Each copy of s must provide the same natural

Example: A counting server

Goal is to define a server P that provides an increasing series of naturals.

$$\nu s.(P \mid a[b].(b[c].(?s[x].c \leftrightarrow x \mid ?s[y].b \leftrightarrow y) \mid ?s[z].a \leftrightarrow z))$$

Should be able to produce any ordering of 1,2,3 on a.

Recursion likely insufficient

- Can provide distinct streams of naturals on contraction
- But produced uniformly, while usage of contracted streams need not be uniform

Conclusions

- Adding fixed points
  - Allows (some) recursive sessions
  - More expressive servers
  - Still race- and deadlock-free
- Still less expressive than we'd like
- Doesn't address non-determinism

Conclusions

- Adding fixed points
  - Allows (some) recursive sessions
  - More expressive servers
  - Still race- and deadlock-free
- Still less expressive than we'd like
- Doesn't address non-determinism

Thank you!

Additional slides

Proof rules for fixed points

Two new proof rules

$$[\mu]\ \frac{P \vdash x : B\{\mu X.B/X\}, \Gamma}{P \vdash \mathbf{unr}\, x.P \vdash x : \mu X.B, \Gamma}$$

$$[\nu]\ \frac{P \vdash y : A, \Gamma \quad Q \vdash y : A^{\perp}, x : B\{A/X\}}{\mathbf{roll}\ x\langle y\rangle(P, Q) \vdash \nu X.B}$$