Stratified Bounded Affine Logic
for Logarithmic Space

Ulrich Schöpp
University of Munich
Programming with Logarithmic Space

Goal: *Find a programming language for the functions computable in logarithmic space.*
Programming with Logarithmic Space

Goal: *Find a programming language for the functions computable in logarithmic space.*

Stratified Bounded Affine Logic (SBAL)

- Type system for a functional language with
  - higher-order functions
  - polymorphism
  - inductive datatypes

- Well-typed programs represent the functions computable in logarithmic space
Linear Logic and Complexity

Capture complexity classes as sub-systems of System F by restricting Intuitionistic Affine Logic ($\otimes$, $\rightarrow$, $!$, $\forall$).

**Polynomial time**
- *Bounded Linear Logic* — ! bounded by polynomial
- *Light Linear Logic* — $!,\exists$
- *Soft Linear Logic* — multiplexing only

**Elementary functions**
- *Elementary Affine Logic* — no digging

...  

**Logarithmic space**
- *Stratified Bounded Affine Logic* — restrict ! and $\forall$
Bounded Affine Logic

- Bounded Linear Logic [Girard, Scedrov & Scott 1989]
- Affine variant [Hofmann & Scott 2000]
- BAL = Affine logic with bounded modality

Affine Logic

\[ !A \quad \text{— arbitrarily many copies of } A \]

Bounded Affine Logic

\[ !_{x<p}A(x) \quad \text{— } p \text{ many copies of } A \]

\[ !_{x<p}A(x) = A(0) \otimes A(1) \otimes \cdots \otimes A(p - 1) \]
Bounded Affine Logic

- Formulae

\[ A ::= \alpha(p_1, \ldots, p_n) \mid A \otimes A \mid A \rightarrow A \mid !_{x<p} A \mid \forall \alpha. A \]

- Rules for \( \otimes \), \( \rightarrow \), \( \forall \) essentially as in Affine Logic

\[
\begin{align*}
\text{Affine Logic} & \quad \text{Bounded Affine Logic} \\
\Gamma, \alpha & \vdash \alpha & p \leq q \\
\Gamma, A[B/\alpha] & \vdash C & \Gamma, \alpha(p) \vdash \alpha(q) \\
\Gamma, \forall \alpha. A & \vdash C & \Gamma, \forall \alpha. A \vdash C
\end{align*}
\]
Bounded Affine Logic – Modality

\[ \frac{\Gamma, A[0/x] \vdash B}{\Gamma, !x < 1+w A \vdash B} \quad \frac{\Gamma, !x < p A, !y < q A[p + y/x] \vdash B}{\Gamma, !x < p+q+w A \vdash B} \]

\[ \frac{\Gamma \vdash A}{!x < p \Gamma \vdash !x < p A} \]

\[ \frac{\Gamma, !x < p !z < q(x) A[z + \sum_{u < x} q(u)/y] \vdash B}{\Gamma, !y < \sum_{x < p} q(x) A \vdash B} \]
Bounded Affine Logic

Natural Numbers
\[ \mathbb{N}_x = \forall \alpha. \forall y < x (\alpha(y) \rightarrow \alpha(y + 1)) \rightarrow \alpha(0) \rightarrow \alpha(x) \]

Binary Lists
\[ \mathbb{L}_x = \forall \alpha. \forall y < x (\alpha(y) \rightarrow \alpha(y + 1)) \rightarrow \alpha(0) \rightarrow \alpha(x) \]

**Theorem:** The proofs of sequents of the form
\[ \mathbb{L}_x \vdash \mathbb{L}_{p(x)} \]
represent exactly the functions in PTIME.
[Girard, Scedrov & Scott ‘91], [Hofmann & Scott 2000]
Stratified Bounded Affine Logic

- BAL = Affine logic with bounded modality
- SBAL = BAL with bounded universal quantifier

Bounded Linear Logic

\[ \forall \alpha. A \quad \text{— unbounded quantifier} \]

Stratified Bounded Linear Logic

\[ \forall \alpha \leq p. A \quad \text{— quantifier bounded by polynomial } p \]
Quantifier Rules

Bounded Affine Logic

\[ \Gamma, A[\lambda \vec{x}. B / \alpha] \vdash C \]
\[ \frac{\Gamma \vdash A}{\Gamma, \forall \alpha. A \vdash C} \]
\[ \alpha \notin \Gamma \]

Stratified Bounded Affine Logic

\[ \Sigma \vdash B \leq p(\vec{x}) \]
\[ \Sigma \mid \Gamma, A[\lambda \vec{x}. B / \alpha] \vdash C \]
\[ \frac{\Sigma \mid \Gamma, \forall \alpha \leq p. A \vdash C}{\Sigma, \alpha \leq p \mid \Gamma \vdash A} \]
\[ \alpha \notin \Gamma \]
Examples

\[ N^p_x = \forall \alpha \leq p. !y<x(\alpha(y) \rightarrow \alpha(y + 1)) \rightarrow \alpha(0) \rightarrow \alpha(x) \]

The following functions are definable as usual:

\[ \vdash \text{zero} : N^p_x \]
\[ \vdash \text{succ} : N^p_x \rightarrow N^p_{x+1} \]
\[ \vdash \text{add} : N^p_x \rightarrow N^q(p) \rightarrow N^p_{x+y} \]
\[ \vdash \text{pred} : N^q(p) \rightarrow N^p_x \]

Coercion:

\[ \vdash \text{coerc} : N^r(p,q) \rightarrow !_p N^q_x \]

BAL: \[ \vdash \text{coerc} : N_x \rightarrow !_p N_x \]
Corollary: Each SBAL-proof of a sequent

\[ L^r_x \vdash L^q_{p(x)} \]

for sufficiently large \( q \) represents a LOGSPACE function.

\[ L^p_x = \forall \alpha \leq p. \ !y<x (\alpha(y) \to \alpha(y + 1)) \to \]
\[ \to !y<x (\alpha(y) \to \alpha(y + 1)) \to \alpha(0) \to \alpha(x) \]
Functional Programs in Logarithmic Space

• Pointer-algorithms

• Representation by higher-order functions

\[
\text{let } \text{cons}_0 (f:\text{nats} \to \{0,1,\text{blank}\}) = \\
\text{fun } (n:\text{nats}) - \text{ if } n=0 \text{ then } 0 \text{ else } f(n-1)
\]

\[
\text{cons}_0 : \{0,1\}^* \to \{0,1\}^*
\]

\[
\text{cons}_0(w) = 0w
\]

• Standard evaluation strategies (e.g. call-by-value) use linear space.
Functional Programs in Logarithmic Space

- Pointer-algorithms
- Representation by higher-order functions

\[
\text{cons}_0 : \{0, 1\}^* \rightarrow \{0, 1\}^*
\]

\[
\text{cons}_0(w) = 0w
\]

let \(\text{cons}_0 (f:\text{nat-} \{0,1,\text{blank}\}) =\)

fun (n:nat) - if n=0 then 0 else f (n-1)

- In SBAL:

\[
\vdash \text{cons}_0 : (\mathbb{N}_x^q \rightarrow 3) \rightarrow (\mathbb{N}_x^p \rightarrow 3)
\]
SBAL – Complexity II

\[ W_{x}^{p,q,r} = (\forall_{q} N_{x}^{p} \rightarrow 3^{q}) \otimes N_{x}^{r} \]

- word encoded by a function
- upper bound on word length

**Theorem:** Each SBAL-proof of \( \forall_{y<t} W_{x}^{q,r,s} \vdash W_{p(x)}^{u,v,w} \) represents a LOGSPACE function.

**Theorem:** Each SBAL-proof of \( N_{x}^{r} \vdash N_{p(x)}^{q} \) represents a LINSPACE function.
Skewed Iteration

\[ \mathbb{N}_x^p = \forall \alpha \leq p. !y < x (\alpha(y) \rightsquigarrow \alpha(y + 1)) \rightsquigarrow \alpha(0) \rightsquigarrow \alpha(x) \]

The following functions are definable as usual:

\[ \vdash zero : \mathbb{N}_x^p \]

\[ \vdash succ : \mathbb{N}_x^p \rightarrow \mathbb{N}_{x+1}^p \]

\[ \vdash add : \mathbb{N}_x^p \rightarrow \mathbb{N}_{y}^{q(p)} \rightarrow \mathbb{N}_{x+y}^p \]

\[ \vdash pred : \mathbb{N}_{x}^{q(p)} \rightarrow \mathbb{N}_x^p \]

Subtraction is not!
Skewed Iteration

- Iteration principle must be implemented without storage of intermediate values, e.g. for $\oplus = L_x$
  
  $n: \forall \alpha \leq p. !y < x (\alpha(y) \rightarrow \alpha(y + 1)) \rightarrow \alpha(0) \rightarrow \alpha(x)$

  $\Rightarrow$ Tail-call optimisation not possible

```
x := g;
while n \neq 0 do (x := f(x); n := n-1)
return x;
```

- Without this optimisation, subtraction (defined by iteration of `pred`) uses more than LOGSPACE.
Skewed Iteration

- Tail-call optimisation possible for small types:

\[ A, B ::= B^p \mid N^p_x \mid A \otimes B \]

- Skewed Iteration
For small types, size-polynomials can be ignored:

\[ N^p_x \rightarrow! y < x (N^q_x \rightarrow N^r_x) \rightarrow N^s_x \rightarrow N^t_x \]

\[ N^p_x \rightarrow! y < x (N^q_x \otimes N^r_x \rightarrow N^s_x \otimes N^t_x) \rightarrow N^u_x \otimes N^v_x \rightarrow \ldots \]

...
SBAL – Complexity III

**Theorem:** Each proof in SBAL+(Skew) of

\[ \forall y < t \forall x \forall q, r, s \exists u, v, w \exists u, v, w \]

represents a LOGSPACE function.

**Theorem:** Each LOGSPACE function can be represented in SBAL+(Skew) in this way.
From SBAL to Logarithmic Space

- BC⁻ [Murawski & Ong], [Møller-Neergaard & Mairson]
- Model computation as an interaction process
  - Recomputation of intermediate values
  - Question/Answer dialogues

⇒ Game Semantics

⇒ Geometry of Interaction Situation (GoI) [Abramsky, Haghverdi & Scott]
A Geometry of Interaction Situation

- **Objects** $A = (A^-, A^+)$
- **Morphisms** $A \rightarrow B$

Partial functions $A^+ + B^- \rightarrow A^- + B^+$
A Geometry of Interaction Situation

- Identity $id: A \rightarrow A$

- Composition of $f: A \rightarrow B$ and $g: B \rightarrow C$

\[ g \cdot f = \]

\[ A^{-} \quad \square \quad A^{+} \quad A^{+} \quad A^{-} \]

\[ C^{-} \quad \square \quad C^{+} \quad B^{-} \quad B^{+} \]

$A^{-} \quad \square \quad A^{+} \quad A^{-} \quad A^{+}$
GoI Structure

- Tensor product
  $$A \otimes B = (A^- + B^-, A^+ + B^+)$$

- Linear function space
  $$A \rightarrow B = (A^+ + B^-, A^- + B^+)$$

- Modality
  $$!A = (\mathbb{N} \times A^-, \mathbb{N} \times A^+)$$
SBAL and GoI

- Identify LOGSPACE-fragment of this GoI Situation
- Size bound on messages
  \[ !_p A = (\{n \mid n < p\} \times A^-, \{n \mid n < p\} \times A^+) \]
  instead of
  \[ !A = (\mathbb{N} \times A^-, \mathbb{N} \times A^+) \]
- Restriction of morphisms to LINSPACE
- Translation of SBAL-typed terms in GoI-programs
- Realisability model based on the BAL-model of [Hofmann & Scott 2000]
Realisability

Interpret SBAL-formula with resource variables in $X$ by

$\left( |A|, \|A\|, \vdash \right)$

System F type

Realising Object

Realisation Relation
Realisability

Interpret SBAL-formula with resource variables in $X$ by

\[ (|A|, \|A\|, \vdash) \]

GoI-Object $\|A\|_{\eta}$ for each $X$-environment $\eta$

Relation $\eta, e \vdash a$

$\eta \in V(X)$, $e : \|A\|_{\eta}^{-} \to \|A\|_{\eta}^{+}$, $a \in |A|$
Realisability

Interpret SBAL-formula with resource variables in $X$ by

$$(|A|, \|A\|, \models)$$

Size condition:

$$\forall \eta \in V(X). \forall x \in \|A\|_\eta^- \cup \|A\|_\eta^+. |x| \leq c \cdot |\eta| + d.$$
Realisability – Morphisms

Interpret SBAL sequent $\Sigma \vdash A \Rightarrow B$ by a System F-term $M: \|A\| \Rightarrow \|B\|$ that has a realiser:

$$r: \prod_{\eta \in V(X)} \|A\|_\eta \rightarrow \|B\|_\eta$$

1. If $\eta, e \vdash a$ then $\eta, r_\eta e \vdash Ma$.

2. The function $\langle \eta, q \rangle \mapsto r_\eta(q)$ is computable in linear space.
Example – Small Numbers

\[
\begin{align*}
|S_x| &= \mathbb{N} \\
\|S_x\|_{\eta}^- &= \{*\} \\
\|S_x\|_{\eta}^+ &= \{n \in \mathbb{N} \mid n \leq \eta(x)\} \\
\eta, e \vdash n &\iff n \leq \eta(x) \land e(*) = n
\end{align*}
\]

\[\text{succ} : S_x \rightarrow S_{x+1}\]

\[
\begin{align*}
\eta : \|S_x\|_{\eta}^+ + \|S_{x+1}\|_{\eta}^- &\rightarrow \|S_x\|_{\eta}^- + \|S_{x+1}\|_{\eta}^+ \\
* &\leftrightarrow * \\
n &\leftrightarrow n + 1
\end{align*}
\]
Modality

\[ |!x<p A| = |A| \]

\[ \|!x<p A\|_\eta = \left( \sum_{n<p[\eta]} \|A\|_{\eta[n/x]}^-, \sum_{n<p[\eta]} \|A\|_{\eta[n/x]}^+ \right) \]
Contraction

\[ !_{x<p+q}A \rightarrow ( !_{x<p}A) \otimes ( !_{y<q}A[p + y/x] ) \]

Domain of realiser:

\[ \sum_{n<p[\eta]+q[\eta]} \| A \|^{+}_{\eta[n/x]} + \left( \sum_{n<p[\eta]} \| A \|^{-}_{\eta[n/x]} + \sum_{n<q[\eta]} \| A \|^{-}_{\eta[n+p[\eta]/x]} \right) \]

Range of realiser:

\[ \sum_{n<p[\eta]+q[\eta]} \| A \|^{-}_{\eta[n/x]} + \left( \sum_{n<p[\eta]} \| A \|^{+}_{\eta[n/x]} + \sum_{n<q[\eta]} \| A \|^{+}_{\eta[n+p[\eta]/x]} \right) \]
Contraction

\[ !x<p+qA \rightarrow ( !x<pA ) \otimes ( !y<qA[p+y/x] ) \]

\[ \sum_{n<p[p]+q[q]} \| A \|^{+}_{\eta[n/x]} + \left( \sum_{n<p[p]} \| A \|^{-}_{\eta[n/x]} + \sum_{n<q[q]} \| A \|^{-}_{\eta[n+p[p]/x]} \right) \]

\[ \sum_{n<p[p]+q[q]} \| A \|^{-}_{\eta[n/x]} + \left( \sum_{n<p[p]} \| A \|^{+}_{\eta[n/x]} + \sum_{n<q[q]} \| A \|^{+}_{\eta[n+p[p]/x]} \right) \]
Complexity

**Theorem:** Each morphism
\[ !_q S_x \rightarrow S_{p(x)} \]
has a LINSPACE-function as its underlying function.

**Theorem:** Each morphism
\[ !_q (S_x \rightarrow S_3) \otimes !_q S_x \rightarrow (S_{p(x)} \rightarrow S_3) \otimes S_{p(x)} \]
represents a LOGSPACE-function.
Universal Quantifier

- Universal quantifiers in realisability models

\[ r \text{ realises } f : \forall \alpha. A \]
\[ \iff \]
\[ r \text{ realises all instances } f \ B : A[B/\alpha] \]

- Realising object?

\[ \| \forall \alpha. \alpha \to \alpha \|_\eta \text{ must be as good as all instances} \]
\[ \| B \to B \|_\eta, \quad \|(B \to B) \to (B \to B)\|_\eta, \]
\[ \|(C \otimes (B \to B)) \to (C \otimes (B \to B))\|_\eta, \ldots \]
Universal Quantifier

- Universal quantifiers in realizability models

\[ r \text{ realises } f : \forall \alpha. A \]

\[ \iff \]

\[ r \text{ realises all instances } f B : A[B/\alpha] \]

- Realising object?

\[ \| \forall \alpha. \alpha \to \alpha \|_\eta \text{ must be as good as all instances} \]

\[ \| \forall \alpha. \alpha \to \alpha \|_\eta = (\mathbb{N}, \mathbb{N}) \to (\mathbb{N}, \mathbb{N}) \]
Universal Quantifier

- Universal quantifiers in realisability models

\[ r \text{ realises } f : \forall \alpha. A \]

Condition:
\[ \forall \eta \in V(X). \forall x \in \|A\|^- \cup \|A\|^+. |x| \leq c \cdot |\eta| + d. \]

\[ \|\forall \alpha. \alpha \rightarrow \alpha\|_\eta \] must be as good as all instances

\[ \|\forall \alpha. \alpha \rightarrow \alpha\|_\eta = (\mathbb{N}, \mathbb{N}) \rightarrow (\mathbb{N}, \mathbb{N}) ? \]
Universal Quantifier

- Restrict the range of the quantifier to objects whose questions and answers can be encoded within a certain bound.

\[ \| \forall \alpha \leq p. \alpha \rightarrow \alpha \|_{\eta} = (N_{p[\eta]}, N_{p[\eta]}) \rightarrow (N_{p[\eta]}, N_{p[\eta]}) \]
Conclusion

SBAL is a functional programming for LOGSPACE with higher-order functions, polymorphism and inductive datatypes.

Further Work

- Simplification without resource polynomials, perhaps as in Light Affine Logic
- Logical status of skewed iteration
- Structured data (e.g. graphs)