Overview of Feedback, and the Feedback Hyperjump

Robert Lubarsky, Florida Atlantic University
includes joint work with Nate Ackerman, Harvard University
and Cameron Freer, Massachusetts Institute of Technology

Logic Colloquium ’19
Prague, Czech Republic
August 12-16, 2019
Let $H_X(e) = \uparrow \text{ resp. } \downarrow \text{ iff } \{e\}^X \uparrow \text{ resp. } \downarrow$. 
Let $H_X(e) = \uparrow$ resp. $\downarrow$ iff $\{e\}^X \uparrow$ resp. $\downarrow$.

Any fixed point of the operator $X \mapsto H_X$ gives a coherent notion of feedback. The easiest semantics is the least fixed point.
Tree of Sub-Computations
Another Good Example
A Bad Example

One can naturally define the course of a computation if and only if the tree of sub-computations is well-founded.
A Bad Example

One can naturally define the course of a computation if and only if the tree of sub-computations is well-founded.
Allow all possible sub-computation calls, even if the tree of sub-computations is ill-founded, and consider only those for which the tree of sub-computations just so happens to be well-founded.
Allow all possible sub-computation calls, even if the tree of sub-computations is ill-founded, and consider only those for which the tree of sub-computations just so happens to be well-founded. So some legal computations have an undefined result: the *freezing* computations. The *non-freezing* computations have a perfectly well-defined semantics.
Allow all possible sub-computation calls, even if the tree of sub-computations is ill-founded, and consider only those for which the tree of sub-computations just so happens to be well-founded. So some legal computations have an undefined result: the freezing computations. The non-freezing computations have a perfectly well-defined semantics.

Notation: $\langle e \rangle(n)$
Feedback Primitive Recursion: Let $h$ be the smallest function such that $h(e, \vec{n}) = \{e\}^h(\vec{n})$, $e$ a code for a primitive recursive function.
Other Kinds of Feedback

- Feedback Primitive Recursion: Let $h$ be the smallest function such that $h(e, n) = \{e\}^h(n)$, $e$ a code for a primitive recursive function.

- Feedback Hyperarithmetic Computability: Consider $X \mapsto \mathcal{O}^X$ (cf. Kleene’s $\mathcal{O}$, or ATR).
  Let $X$ be the smallest function such that $X = \mathcal{O}^X$. 
Other Kinds of Feedback

▶ Feedback Primitive Recursion: Let $h$ be the smallest function such that $h(e, \vec{n}) = \{e\}^h(\vec{n})$, $e$ a code for a primitive recursive function.

▶ Feedback Hyperarithmetic Computability: Consider $X \mapsto O^X$ (cf. Kleene’s $O$, or ATR).
Let $X$ be the smallest function such that $X = O^X$.

▶ Feedback Turing on Cantor Space: Let $f(Y) : C \rightarrow C$ be $\langle e \rangle^Y$. 

Robert Lubarsky
Overview of Feedback, and the Feedback Hyperjump
Other Kinds of Feedback

- Feedback Primitive Recursion: Let $h$ be the smallest function such that $h(e, \bar{n}) = \{e\}^h(\bar{n})$, $e$ a code for a primitive recursive function.

- Feedback Hyperarithmetic Computability: Consider $X \mapsto O^X$ (cf. Kleene's $O$, or ATR).
  Let $X$ be the smallest function such that $X = O^X$.

- Feedback Turing on Cantor Space: Let $f(Y) : \mathcal{C} \rightarrow \mathcal{C}$ be $\langle e \rangle^Y$.

- Parallel Feedback Turing Computability: Allows oracle questions of the form $\{e\}(\cdot)\$, with answer some $\{e\}(n) = k$. 
Theorem

(AFL) $Y$ is feedback Turing computable iff $Y$ is hyperarithmetic iff $Y$ is $\Delta^1_1$ iff $Y \in L_{\omega_1^{CK}}$.

So feedback provides a machine model without higher types for the hyperarithmetic sets.
Theorem

(AFL) $Y$ is feedback Turing computable iff $Y$ is hyperarithmetic iff $Y$ is $\Delta^1_1$ iff $Y \in L^\omega_{\omega_1}^{CK}$.

So feedback provides a machine model without higher types for the hyperarithmetic sets.

Theorem

(AFL) $Y$ is feedback primitive recursive iff $Y$ is partial computable.
Theorem

(AFL) \( Y \) is feedback Turing computable iff \( Y \) is hyperarithmetic iff \( Y \) is \( \Delta^1_1 \) iff \( Y \in L_{\omega_1^{CK}} \).

So feedback provides a machine model without higher types for the hyperarithmetic sets.

Theorem

(AFL) \( Y \) is feedback primitive recursive iff \( Y \) is partial computable.

Theorem

(AFL) \( f : C \to C \) is feedback computable iff \( f \) is \( \Delta^1_1 \) (\( f \) is Borel).
Theorems

**Theorem**

(AFL) \( Y \) is feedback Turing computable iff \( Y \) is hyperarithmetic iff \( Y \) is \( \Delta_1 \) iff \( Y \in L_{\omega_1^{CK}} \).

So feedback provides a machine model without higher types for the hyperarithmetic sets.

**Theorem**

(AFL) \( Y \) is feedback primitive recursive iff \( Y \) is partial computable.

**Theorem**

(AFL) \( f : C \rightarrow C \) is feedback computable iff \( f \) is \( \Delta_1 \) (\( f \) is Borel).

**Theorem**

(L) \( Y \) is parallel feedback Turing computable iff \( Y \in L_\gamma \), where \( \gamma \) is the least ordinal which is \( \Pi_1 \) gap-reflection on admissibles.
Gap-Reflection

Definition
Given \( \delta \), consider \( \phi(\delta) \) a \( \Pi_1 \) sentence with parameters \( \delta \) and members of \( L_\delta \). Then \( \delta \) is \( \Pi_1 \) gap-reflecting on admissibles if for all such \( \phi \), if \( L_\delta^+ \models \phi(\delta) \), then for some \( \beta < \delta \) \( L_\beta^+ \models \phi(\beta) \).
Gap-Reflection

Definition

Given $\delta$, consider $\phi(\delta)$ a $\Pi_1$ sentence with parameters $\delta$ and members of $L_\delta$. Then $\delta$ is $\Pi_1$ gap-reflecting on admissibles if for all such $\phi$, if $L_{\delta^+} \models \phi(\delta)$, then for some $\beta < \delta$ $L_{\beta^+} \models \phi(\beta)$.

Best example: $\psi(\delta) = T_\delta$ has an ordinal ranking function, $T_\delta$ a tree definable from $\delta$. 
Gap-Reflection

Definition

Given $\delta$, consider $\phi(\delta)$ a $\Pi_1$ sentence with parameters $\delta$ and members of $L_\delta$. Then $\delta$ is $\Pi_1$ **gap-reflecting on admissibles** if for all such $\phi$, if $L_{\delta^+} \models \phi(\delta)$, then for some $\beta < \delta$ $L_{\beta^+} \models \phi(\beta)$.

Best example: $\psi(\delta) = T_\delta$ has an ordinal ranking function, $T_\delta$ a tree definable from $\delta$.

The least such ordinal, $\gamma$, is

- the least $\Sigma^1_1$-reflecting ordinal,
- the closure of $\Sigma_2$ definable sets in the $\mu$-calculus,
- the closure of $\Sigma^1_1$ monotone inductive definitions,
- the least ordinal over which $\Sigma^0_2$ Determinacy holds, and
- the least ordinal with the $\Sigma^1_1$ Ramsey property.
A natural number $n$ induces a tree of ordinal notations $T_n$. 
Kleene’s $\mathcal{O}$

A natural number $n$ induces a **tree of ordinal notations** $T_n$. $n \in \mathcal{O}$ iff $T_n$ is (well-formed and) well-founded.
Relativize: Let $H_X(n) = \downarrow$ resp. $\uparrow$ iff $T^X_n$ is well- resp. ill-founded, where $T^X_n$ must be fully defined (i.e. not freezing). Let $SO$ be the least fixed point.
Relativize: Let \( H_X(n) = \downarrow \) resp. \( \uparrow \) iff \( T_n^X \) is well- resp. ill-founded, where \( T_n^X \) must be fully defined (i.e. not freezing).

Let \( SO \) be the least fixed point.

Conjecture/Theorem: A set is computable from \( SO \) iff it is in \( L_\alpha \), where \( \alpha \) is the least recursively inaccessible.
Let $H_X(n) = \downarrow$ iff $T^X_n$ is well-founded, where $T^X_n$ must be non-freezing, and $H_X(n) = \uparrow$ iff $T^X_n$ is ill-founded (even if $T^X_n$ is freezing, a kind of tree parallelism). Let $LO$ be the least fixed point.
Let $H_X(n) = \Downarrow$ iff $T_n^X$ is well-founded, where $T_n^X$ must be non-freezing, and $H_X(n) = \Uparrow$ iff $T_n^X$ is ill-founded (even if $T_n^X$ is freezing, a kind of tree parallelism).

Let $\mathcal{L}O$ be the least fixed point.

Conjecture/Theorem: A set is computable from $\mathcal{L}O$ iff it is in $L_\gamma$, where $\gamma$ is the least ordinal which is $\Pi_1$ gap-reflecting on admissibles.
Least fixed point semantics for other kinds of computability, such as:

- $K_2$ computability,
- E-recursion,
- Lifschitz computability,
- infinitary and register machines,
- graph models.
Example

Feedback Turing: Recall the monotone inductive operator $H_X(e) = \uparrow$ resp. $\downarrow$ iff $\{e\}^X \uparrow$ resp. $\downarrow$.

Take the least fixed point. Set all freezing computations to “divergent” and iterate $H_X$ to a fixed point. Repeat, until you have a fixed point of that operation. What does that compute?
Iterated Feedback

Example

Let a second oracle tell you when computations relative to the first oracle are freezing (level 0 and level 1 freezing). What does that compute?
Iterated Feedback

Example
Let a second oracle tell you when computations relative to the first oracle are freezing (level 0 and level 1 freezing). What does that compute?

Example
Iterate levels of freezing along any ordinal. What does that compute?
Iterated Feedback

Example
Let a second oracle tell you when computations relative to the first oracle are freezing (level 0 and level 1 freezing). What does that compute?

Example
Iterate levels of freezing along any ordinal. What does that compute?

Example
Iterate levels of freezing along a ordinal built dynamically during the computation. What does that compute?
Feedback along an Order

Example
Extend the definition of iteration along an ordinal to iteration along any partial order. For interesting partial orders (e.g. the rationals), what does that compute? What does this compute along any partial order built dynamically?
Feedback along an Order

Example
Extend the definition of iteration along an ordinal to iteration along any partial order. For interesting partial orders (e.g. the rationals), what does that compute? What does this compute along any partial order built dynamically?

Example
Extend the definition of iteration along a partial order to iteration along an order. For instance, two oracles, each of which gives freezing information about the other. What does that compute? What kind of information does that yield?
Nathanael Ackerman, Cameron Freer, and Robert Lubarsky, “Feedback Turing Computability, and Turing Computability as Feedback,” *Proceedings of LICS 2015*, Kyoto, Japan


Robert Lubarsky, “Parallel Feedback Turing Computability,” *Proceedings of LFCS ’16, LNCS 9537*