A Typed Language for Truthful One-Dimensional Mechanism Design

Andrei Lapets Alex Levin

We leverage programming language design techniques in creating a typed language for describing mechanisms in which well-typed expressions are necessarily truthful mechanisms.

Theorem 1. *A mechanism $M = (A, p)$ is* ***truthful*** *if and only if its allocation algorithm A is* monotone *and it collects as payment the* critical value *from every bidder.*

Definitions

- Bidders: $i, j \in \{1, \ldots, n\}$
- Private values (and bids): $v \in \mathbb{R}_+^n$, $v_i \in \mathbb{R}_+$
- Outcomes: $o \in \mathcal{O}$, $o_i \in \{0, 1\}$
- Allocation algorithms: $A \in \mathbb{R}_+^n \to \mathcal{O}$
- Welfare: $w_o(v) = \sum_i o_i v_i$
- Payment functions: $p(v) \in \mathbb{R}_+^n$
- Mechanisms: $M = (A, p)$

Definition 1. An A is **monotone** if $\forall j \forall v_{-j}$,

$$\forall v_j' \geq v_j,\ A_j(v_{-j}, v_j) = 1 \Rightarrow A_j(v_{-j}, v_j') = 1.$$

Definition 2. [MN02] A monotone A is **bitonic** if $\forall j$, $\forall v_{-j}$, $w_{A(v)}(v_{-j}, v_j)$ is non-increasing for $v_j < \theta_j(v_{-j})$, non-decreasing for $v_j \geq \theta_j(v_{-j})$.

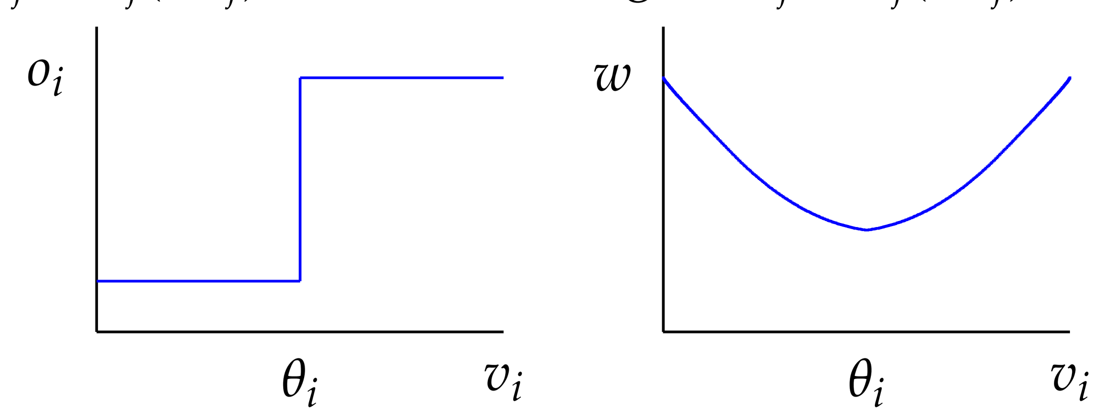

Figure 1: Monotonicity and bitonicity.

- A monotone A has for every bidder j a critical value $\theta_j(v_{-j}) \in \mathbb{R} \cup \{\infty\}$.
- Quasi-linear utility: bidder j with value v_j has utility $v_j A_j(v_j', v_{-j}) - p_j(v_j', v_{-j})$ given mechanism $M = (A, p)$ and bids (v_j', v_{-j}).

Definition 3. Mechanism $M = (A, p)$ is truthful when $\forall v_j \forall v_{-j}$, for all $v_j' \neq v_j$,

$$v_j A_j(v_j, v_{-j}) - p_j(v_j, v_{-j}) \geq v_j A_j(v_j', v_{-j}) - p_j(v_j', v_{-j}).$$

Decision Trees

We first consider algorithms which are decision trees with comparisons between bid values at the nodes (e.g. $v_4 \geq v_7$), and allocations to a single agent at the leaves.

$$\begin{array}{rrcl}
\text{natural} & i & \in & \mathbb{N} \\
\text{bid vector} & v & \in & \mathbb{R}^n \\
\text{primitive} & p & ::= & \texttt{alloc} \mid \texttt{value} \\
\text{expression} & e & ::= & i \mid v \mid p \mid e_1\ e_2 \\
& & \mid & \texttt{if}\ e_1 \geq e_2\ \texttt{then}\ e_3\ \texttt{else}\ e_4
\end{array}$$

Monotonicity can be checked efficiently through a syntax-directed analysis of the algorithm. For each bidder, every possible execution path is analyzed, and the critical intervals under every execution path are constructed.

Theorem 2. *A decision tree e represents a monotonic allocation algorithm iff for every bidder, under every execution path, the critical interval can be represented as a critical value threshold function.*

Example: Two-bidder VCG

Suppose we have an algorithm which allocates to one of two bidders with higher value:

```
if value 1 v >= value 2 v then
  alloc 1 v
else
  alloc 2 v
```

For bidder 1, this is first converted into an expression on intervals over $\mathbb{R}$:

$$([0, \infty) \cap [v_2, \infty)) \cup (\emptyset \cap [0, v_2)).$$

Then, according to the reduction rules for intersection and union, this reduces to:

$$[v_2, \infty),$$

A similar approach can be used to determine that the critical interval for bidder 2 is

$$(v_1, \infty).$$

Since both critical intervals can be represented using a critical value threshold function, we know the allocation algorithm is monotonic.

General Algorithms

We start with a simply-typed λ-calculus with domain-specific primitives and type annotations.

$$\begin{array}{rrcl}
\text{primitive} & p & ::= & \texttt{alloc} \mid \texttt{value} \mid \texttt{max} \mid \texttt{+} \mid \geq \mid \ldots \\
\text{expression} & e & ::= & \textit{bool} \mid \textit{real} \mid i \mid v \mid p \\
& & \mid & e_1\ e_2 \mid \texttt{if}\ e_1\ \texttt{then}\ e_2\ \texttt{else}\ e_3 \\
& & \mid & x \mid \texttt{arg}\ x.e \mid \texttt{fix} \\
\text{proposition} & P & ::= & \text{Biton} \mid \text{Mon} \mid \text{Indep} \\
\text{base type} & \varsigma & ::= & \texttt{Bool}_P \mid \mathbb{N}_P \mid \mathbb{R}_P \mid V \mid \mathcal{O}_P \\
\text{type} & \tau & ::= & \varsigma \mid \tau \to \tau
\end{array}$$

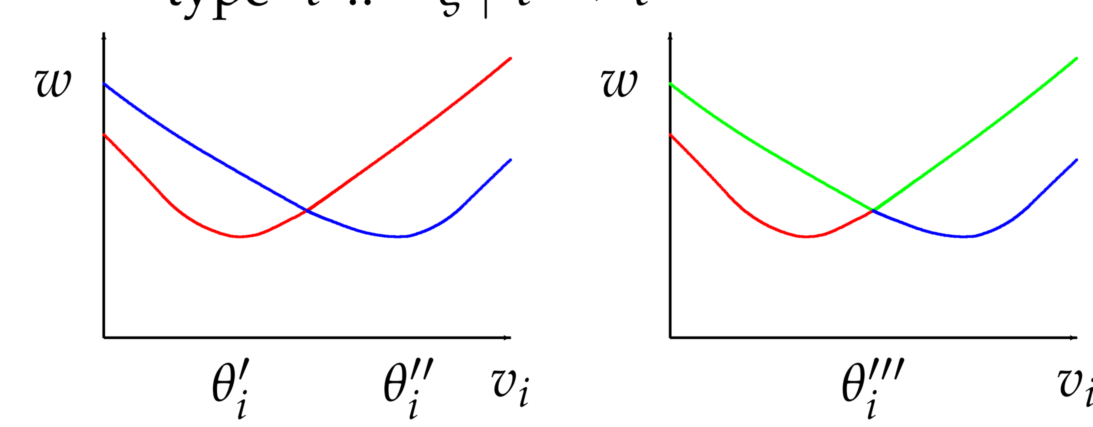

Figure 2: Maximum of bitonic functions.

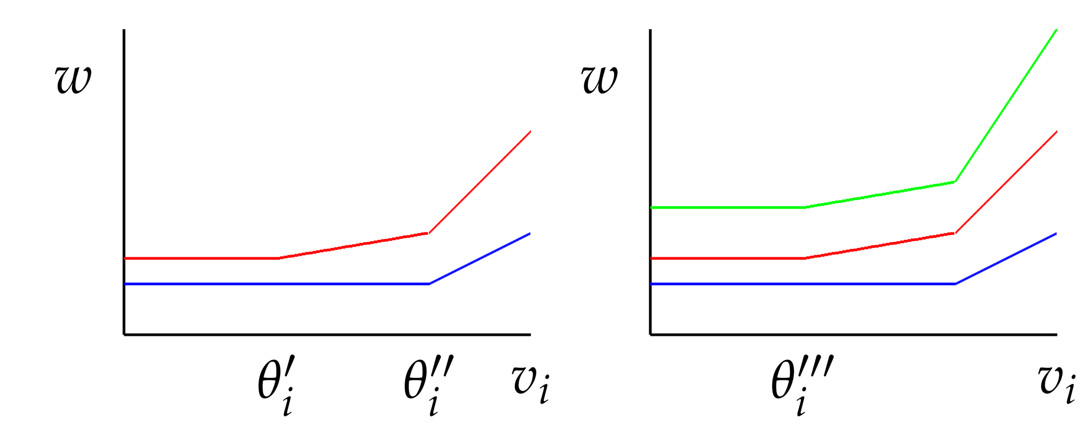

Figure 3: Sum of constant-monotonic functions.

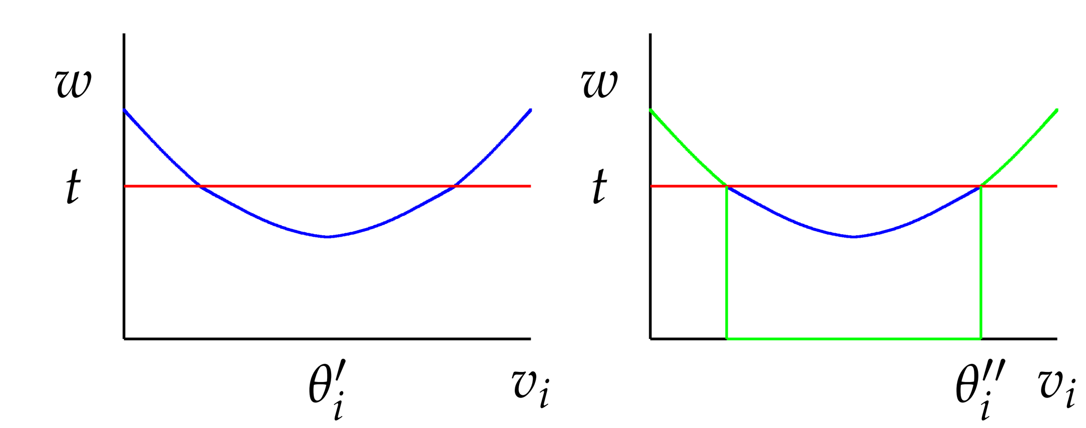

Figure 4: Constant threshold on bitonic function.

We introduce primitives for manipulating outcomes, and assign types to them.

$$\begin{array}{rcl}
\texttt{max} & : & \mathcal{O}_{\text{Biton}} \to \mathcal{O}_{\text{Biton}} \to \mathcal{O}_{\text{Biton}} \\
\texttt{combine} & : & \mathcal{O}_{\text{Biton}} \to \mathcal{O}_{\text{Biton}} \to \mathcal{O}_{\text{Biton}} \\
\texttt{thresh} & : & \mathbb{R}_{\text{Indep}} \to \mathcal{O}_{\text{Biton}} \to \mathcal{O}_{\text{Biton}}
\end{array}$$

These type annotations are induced by the operations on functions: maximum, addition, and threshold.

Example: Exhaustive Search

Mu'alem and Nisan [MN02] define Exst_k, which exhaustively searches all feasible outcomes which allocate to at most k bidders.

```
Exst_k k v = maxAllk v 1 n 1 k noalloc

maxAllk = arg v i n d k o.
  if (or (>= k d) (>= n i)) then
    max
      // try remaining branches 'i' up to 'n'
      (maxAllk v (+ i 1) n d k o)
      // add to accumulator, go deeper
      (maxAllk v 1 n (+ d 1)
                     k (combine (alloc i v) o))
  else if (feasible o) then o else noalloc
```

Example: Profit Extraction

Goldberg et al. [GHK+06] define an algorithm assuming an unlimited supply of an item.

```
filter = arg k v n i.
  if (>= n i) then
    combine (thresh k (alloc i v))
            (filter k v n (+ i 1))
  else noalloc

profitExtract = arg R v n k maxk.
  if (>= maxK k) then
    max (thresh R (filter k v n 1))
        (profitExtract R v n (+ k k) maxk)
  else noalloc
```

For each k, an outcome must have at least R/k bidders with value above k, so the R/kth highest bidder must have value above k.

References

[GHK+06] Andrew Goldberg, Jason Hartline, Anna Karlin, Mike Saks, and Andrew Wright. Competitive auctions. *Games and Economic Behavior*, 55:242–269, 2006.

[MN02] A. Mu'alem and N. Nisan. Truthful approximation mechanisms for restricted combinatorial auctions. In *Proc. 18th National Conference on Artificial Intelligence (AAAI-02)*, 2002.

Advisors: David Parkes , Assaf Kfoury