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:** $A \in \mathcal{R}_+^n \rightarrow \mathcal{O}$
- **Welfare:** $w_i(v) = \sum o_i v_i$
- **Payment functions:** $p(v) \in \mathbb{R}_+^n$
- **Mechanisms:** $M = (A, p)$

**Definition 1.** An $A$ is monotone if for all $v \in \mathbb{R}^n_+$, $v_i \in \mathbb{R}_+$

$$A(v_{-j}, v_j) = 1 \Rightarrow A(v_{-j}, v'_j) = 1.$$

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

**Decision Trees**

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

- **Natural $i \in \mathbb{N}$**
- **Bid vector $v \in \mathbb{R}^n_+$**
- **Primitive $p$**

$$p ::= alloc \mid value \mid \max \mid \geq \mid \ldots$$

- **Expression $e$**

$$e ::= i \mid v \mid p \mid e_1 \cdot e_2 \mid \text{if } e_1 \text{ then } e_2 \text{ else } e_3 \mid x \arg \_x \_y \_z$$

**Theorem 1.** A decision tree $e$ represents a monotonic allocation algorithm if 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 $v$ $\geq$ value 2 $v$ then
- alloc 1 $v$
- alloc 2 $v$

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

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

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

$$[v_2, \infty)$$

A similar approach can be used to determine the critical interval for bidder 2 is $[v_1, \infty)$.

**General Algorithms**

We start with a simply-typed $\lambda$-calculus with domain-specific primitives and type annotations.

- **Primitive $p$**

$$p ::= alloc \mid value \mid \max \mid \geq \mid \ldots$$

- **Expression $e$**

$$e ::= i \mid v \mid p \mid e_1 \cdot e_2 \mid \text{if } e_1 \text{ then } e_2 \text{ else } e_3 \mid x \arg \_x \_y \_z$$

**Example:** Exhaustive Search

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

**Example:** Profit Extraction

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

**References**
