Bidding Languages
Professors Greenwald and Oyakawa
2017-03-22

We describe how a bid and valuation may be represented.

1 Introduction

In a combinatorial auction with \( m \) goods, a bid that maps any subset of goods \( G \) to a real number, \( b : 2^G \rightarrow \mathbb{R} \), may require an exponential amount of space, as there are \( 2^m \) possible subsets of \( G \). In this set of notes, we go over how bids can be implemented using bidding languages. While called bidding languages, and can be used to transmit bid information in an auction setting, we can use bidding languages to describe valuations. The language described here is quite simple, and contains only two basic operations: OR and XOR.

2 The OR/XOR Bidding Language

The high-level idea behind the OR/XOR bidding language\(^1\) is fairly simple: we use OR and XOR operations on atomic bids to describe a valuation function, and use the same operations OR and XOR operations on valuation functions. The language will be general enough to describe any monotone and normalized bid and valuation.

**Definition 2.1** (Monotone). A valuation (bid) is said to be monotone if the value of a superset is at least as large as the value of a set:

\[
v(S) \leq v(T), \quad \forall S \subseteq T \subseteq G.
\]  

(1)

**Definition 2.2** (Normalized). A valuation (bid) is said to be normalized if the value of the emptyset is zero:

\[
v(\emptyset) = 0.
\]

(2)

We first describe what an atomic bid is. Then, we describe the XOR and OR operations that act on atomic bids. Finally, we describe the XOR and OR operations applied to valuation functions.

**Definition 2.3** (Atomic bid). An atomic bid is a tuple \((S, p)\), which describes the following valuation:

\[
v(X) = \begin{cases} 
    p & \text{if } S \subseteq X \subseteq G \\
    0 & \text{otherwise.}
\end{cases}
\]

(3)

**Example 2.4.** We have an atomic bid \((\{a, b\}, 10)\) to describe valuations for a set of goods \(\{a, b, c\}\).
• The valuation of \( \{a, b, c\} \) is 10 because \( \{a, b\} \subseteq \{a, b, c\} \). That is, \( \{a, b\} \cap \{a, b, c\} = \{a, b\} \).

• The valuation of \( \{a, c\} \) is 0 because \( \{a, b\} \not\subseteq \{a, c\} \). That is, \( \{a, b\} \cap \{a, c\} = \{a\} \neq \{a, b\} \).

• The valuation of \( \emptyset \) is 0 because \( \{a, b\} \not\subseteq \emptyset \). That is, \( \{a, b\} \cap \emptyset = \emptyset \neq \{a, b\} \).

**Remark 2.5.** Any valuation that can be described by a single atomic bid is called a *single-minded* valuation. In general, when every bidder has single-minded valuation, solving for the welfare maximizing allocation is NP-hard.

We can combine atomic bids using the XOR operation. This is not XOR in the traditional logic sense, where the output is true if there are an odd number of true inputs. In this setting, you can think of XOR as a max function: the value of a bundle \( S \) defined by an XOR bid will be determined by the atomic bid \((S_a, p_a)\) with the largest \( p_a \), where \( S \subseteq S_a \).

**Definition 2.6 (XOR bid).** Given atomic bids \((S_a, p_a)\) for \( 1 \leq a \leq A \), an XOR bid for a set of goods is

\[
(S_1, p_1) \text{ XOR } (S_2, p_2) \text{ XOR } \ldots \text{ XOR } (S_A, p_A)
\]

(4)

We use the notation \( \oplus \) and \( \bigoplus \) to describe an XOR bid:

\[
(S_1, p_1) \text{ XOR } (S_2, p_2) \text{ XOR } \ldots \text{ XOR } (S_A, p_A) = (S_1, p_1) \oplus (S_2, p_2) \oplus \ldots \oplus (S_A, p_A)
\]

(5)

\[
= \bigoplus_{a=1}^{A} (S_a, p_a).
\]

(6)

The valuation of bundle \( S \subseteq G \) is the largest \( p_a \) such that \( S_a \subseteq S \):

\[
v(S) = \left( \bigoplus_{a=1}^{A} (S_a, p_a) \right)(S)
\]

(7)

\[
= \max_{a: S_a \subseteq S} p_a.
\]

(8)

**Example 2.7.** We have an XOR bid \((\{a\}, 4) \oplus (\{b, c\}, 5)\).

• The valuation of \( \{a, c\} \) is 4, because \( \{a\} \subseteq \{a, c\} \) and \( \{b, c\} \not\subseteq \{a, c\} \).

• The valuation of \( \{a, b, c\} \) is 5, because \( \{a\} \subseteq \{a, b, c\} \), \( \{b, c\} \subseteq \{a, b, c\} \), and \( v(\{b, c\}) = 5 > 4 = v(\{a\}) \).

• The valuation of \( \{c, d\} \) is 0 because \( \{a\} \not\subseteq \{c, d\} \) and \( \{b, c\} \not\subseteq \{c, d\} \).
The XOR operation is sufficient to describe any monotone and normalized bid or valuation. However, this may take an exponential number of atomic bids. We can potentially reduce the number of atomic bids needed to describe a bid using the OR operation, which you can think of as “add”.

**Definition 2.8 (OR bid).** Given atomic bids \((S_a, p_a)\) for \(1 \leq a \leq A\), an OR bid for a set of goods is

\[
(S_1, p_1) \text{ OR } (S_2, p_2) \text{ OR } \ldots \text{ OR } (S_A, p_A). \tag{9}
\]

We use the notation \(\lor\) and \(\bigvee\) to describe an OR bid:

\[
(S_1, p_1) \text{ OR } (S_2, p_2) \text{ OR } \ldots \text{ OR } (S_A, p_A) = (S_1, p_1) \lor (S_2, p_2) \lor \ldots \lor (S_A, p_A) \tag{10}
\]

\[
= \bigvee_{a=1}^{A} (S_a, p_a). \tag{11}
\]

The valuation of bundle \(S \subseteq G\) is the largest sum of \(p_a\)s, where, for each \(p_i\) and \(p_j\) included in the sum, \(S_i \cap S_j = \emptyset\), \(S_i \subseteq S\) and \(S_j \subseteq S\):

\[
v(S) = \left( \bigvee_{a=1}^{A} (S_a, p_a) \right)(S) \tag{12}
\]

\[
= \max \sum \{p_i \mid S_i \cap S_j = \emptyset, S_i \subseteq S, \forall 1 \leq i \neq j \leq A\}. \tag{13}
\]

**Example 2.9.** We have an OR bid \((\{a\}, 4) \lor (\{b, c\}, 5)\).

- The valuation of \(\{a, c\}\) is 4, because \(\{a\} \subseteq \{a, c\}\) and \(\{b, c\} \not\subseteq \{a, c\}\).

- The valuation of \(\{a, b, c\}\) is 9. Partition \(\{a, b, c\}\) into two subsets, \(\{a\}\) and \(\{b, c\}\). The valuation of \(\{a\}\) is 4, and the valuation of \(\{b, c\}\) is 5. Summing them together yields 9.

- The valuation of \(\{c, d\}\) is 0 because \(\{a\} \not\subseteq \{c, d\}\) and \(\{b, c\} \not\subseteq \{c, d\}\).

### 2.1 Expressiveness and Computation

We can convert an OR expression to XOR by considering all combinations of subsets expressed by OR. For example:

\[
(\{a\}, p_a) \lor (\{b\}, p_b) = (\{a\}, p_a) \oplus (\{b\}, p_b) \oplus (\{a, b\}, p_a + p_b) \tag{14}
\]

Converting an OR expression to an XOR expression means we would require an exponential number of atomic bids to represent what OR can succinctly express. However, the advantage of expressing
valuations in terms of XOR is that determining the valuation of a subset of goods $S$ is straightforward, and is described in Algorithm 1.

Similarly, we can convert an XOR expression to OR by using *dummy items*. Let $d$ be a dummy item. Then, given an XOR expression, we can describe it as

$$ (S_a, p_a) \oplus (S_b, p_b) = (S_a \cup \{d\}, p_a) \lor (S_b \cup \{d\}, p_b) \quad (15) $$

Notably, when converting expressions to using OR exclusively, we have the following theorem:

**Theorem 2.10.** Let $s$ be the number of atomic bids in a valuation described by OR and XOR. Such a valuation can be converted to using exclusively OR bids of size $s$ using $O(s^2)$ dummy items.

Thus, there is an advantage to using OR: the size of a bid need not blow up exponentially. However, there is a clear disadvantage. When given the XOR expression

$$ v = (S_1, p_1) \oplus \cdots \oplus (S_n, p_n), \quad (16) $$

we can very quickly compute what $v(S)$ is. All we need to do is iterate through the atomic bids, keeping track of what the largest $p_j$ is when an $S_j \subseteq S$ is encountered. With OR, on the other hand, we are faced with something akin to winner determination, which, in general, is NP-hard. Determining the value of an OR expression is the same as solving the following mathematical program:

$$ \max \sum_{i=1}^{n} v_i(T_i) \quad v_i \text{ is the } i\text{th atomic bid} \quad (17) $$

subject to

$$ T_i \subseteq S_i \quad \forall i \in \{1, \ldots, n\} \quad (18) $$

$$ T_i \cap T_j = \emptyset \quad \forall i \neq j \in \{1, \ldots, n\} \quad (19) $$

$$ T_i \supseteq S_i \quad \forall i \in \{1, \ldots, n\} \quad (20) $$

While bidding languages give us a compact way of expressing valuations, the issue of computation has not disappeared. Expressive languages that allow us to describe valuation functions of interest can shift the burden of space complexity to time complexity.
Notice that if a value of a set of goods cannot be determined in polynomial time with a given bid language, then the auctioneer must spend an exponential amount of time to solve winner determination. Thus, we now shift our focus to settings where we know how to, in polynomial time, calculate a valuation with a given bid.

3 Simple Atomic Bids

We now turn our attention to simple atomic bids, ones which describe the valuation of just a single good. Using singletons, we will show that we can describe several valuations classes of interest.

Definition 3.1 (Singleton). A singleton is an atomic bid \((S, p)\) where \(|S| = 1\).

Example 3.2. We have an singleton \((\{a\}, 10)\) to describe valuations for a set of goods \(\{a, b, c\}\).

- The valuation of \(\{b\}\) is 0 because \(\{a\} \subseteq \{b\}\). That is, \(\{a\} \cap \{b\} = \emptyset\).
- The valuation of \(\{a, c\}\) is 10 because \(\{a\} \subseteq \{a, c\}\). That is, \(\{a\} \cap \{a, c\} = \{a\}\).
- The valuation of \(\emptyset\) is 0 because \(\{a\} \subseteq \emptyset\). That is, \(\{a\} \cap \emptyset = \emptyset \neq \{a\}\).

As usual, we can apply XOR to singletons.

Example 3.3. We have an XOR bid \((\{a\}, 4) \oplus (\{b\}, 5)\).

- The valuation of \(\{a, c\}\) is 4, because \(\{a\} \subseteq \{a, c\}\) and \(\{b\} \subseteq \{a, c\}\).
- The valuation of \(\{a, b, c\}\) is 5, because \(\{a\} \subseteq \{a, b, c\}\), \(\{b\} \subseteq \{a, b, c\}\), and \(v(\{b\}) = 5 > 4 = v(\{a\})\).
- The valuation of \(\{c, d\}\) is 0 because \(\{a\} \subseteq \{c, d\}\) and \(\{b\} \subseteq \{c, d\}\).

Notice that when we use XOR on singletons, the valuation of a bundle is the highest valuation of any good in the bundle. This means that we can describe unit-demand valuations with singletons and XOR operations.

Example 3.4 (Unit-demand valuation). Let a singleton \((\{g\}, p_g)\) describe the valuation of good \(g\), so that \(v(\{g\}) = p_g\) for all \(g \in G\). If we have a unit-demand valuation function, the valuation of a set of goods \(S \subseteq G\) is

\[
v(S) = \max_{g \in S} v(\{g\})
\]
With $m$ singletons, we can describe a unit-demand valuation function.

We can also apply OR to singletons.

**Example 3.5.** We have an OR bid $(\{a\}, 4) \lor (\{b\}, 5)$.

- The valuation of $\{a, c\}$ is 4, because $\{a\} \subseteq \{a, c\}$ and $\{b\} \not\subseteq \{a, c\}$.
- The valuation of $\{a, b, c\}$ is 9. Partition $\{a, b, c\}$ into three subsets, $\{a\}$, $\{b\}$ and $\{c\}$. The valuation of $\{a\}$ is 4, the valuation of $\{b\}$ is 5, and the valuation of $\{c\}$ is 0. Summing them together yields 9.
- The valuation of $\{c, d\}$ is 0 because $\{a\} \not\subseteq \{c, d\}$ and $\{b\} \not\subseteq \{c, d\}$.

Notice that when we use OR on singletons, the valuation of a bundle is the sum of the valuations of the goods in the bundle. This means that we can describe additive valuations with singletons and OR operations.

**Example 3.6 (Additive valuation).** Let a singleton $(\{g\}, p_g)$ describe the valuation of good $g$, so that $v(\{g\}) = p_g$ for all $g \in G$. If we have an additive valuation function, the valuation of a set of goods $S \subseteq G$ is

$$v(S) = \sum_{g \in S} v(\{g\})$$

$$= \left( (\{g_1\}, p_{g_1}) \lor \cdots \lor (\{g_m\}, p_{g_m}) \right)(S)$$

$$= \left( \bigvee_{a=1}^m (\{g_a\}, p_a) \right)(S).$$

With $m$ singletons, we can describe an additive valuation function.

With singletons, determining valuations that use OR operations can be done using Algorithm 2. Thus, in the special case of singletons, we need not require an exponential amount of time to solve the more general winner determination problem if operations were restricted to OR or XOR.

### 4 Combining Functions: OXS and XOS

We now introduce our first combination of operations, where we will apply the OR operation to valuations described by XOR operations. This is called OR-of-XOR. When the atomic bids are singletons, this
Algorithm 2 OR Processing of Singletons

1: for all $g \in S$ do  
   $v_g \leftarrow 0$
3: end for
4: for all $g \in S$ do  
   for $a = 1, \ldots, n$ do  
      if $g \in S_a$ then  
         $v_g \leftarrow \max\{v_g, p_a\}$
8: end if
9: end for
10: end for
11: return $\sum_{g \in S} v_g$

is called OR-of-XOR-of-Singletons, or OXS for short. OR-of-XOR bids looks something like this:

$\left(\cdots \oplus \cdots \oplus \cdots \right) \lor \cdots \lor \left(\cdots \oplus \cdots \oplus \cdots \right)$  \hspace{1cm} (27)

As you can imagine, expressing valuations using atomic bids can take a lot of space. Instead of explicitly writing out atomic bids, we can represent them as functions, and describe XOR and OR in a more compact fashion:

$(u \oplus v)(S) = \max\{u(S), v(S)\}$  \hspace{1cm} (28)

$(u \lor v)(S) = \max_{A,B \subseteq S, A \cap B = \emptyset} u(A) + v(B).$  \hspace{1cm} (29)

Example 4.1. We have an OXS bid $u \lor v$ where $u = ((\{a\}, 4) \oplus (\{b\}, 5))$ and $v = (\{c\}, 7)$.

- The valuation of $\{a, c\}$ is 11. Distribute to $u$ the set $\{a\}$ and to $v$ the set $\{c\}$. $u(\{a\}) = 4$ and $v(\{c\}) = 7$. This means $(u \lor v)(\{a, c\}) = 4 + 7 = 11$.

- The valuation of set $\{b, c\}$ is 12. $u(\{b\})$ is 5 and $v(\{c\})$ is 7.

- The valuation of set $\{a, b, c\}$ is 12. $u(\{a, b\})$ is 5 and $v(\{c\})$ is 7.

A value for any bundle $S \subseteq G$ can be computed in polynomial time using OXS bids. Furthermore, if every bidder submitted OXS bids, then welfare maximization can be done in polynomial time. This is because we can transform a value query into a weighted bipartite graph. For each XOR statement, $X$, construct an edge between $X$ and nodes representing goods. The weight of the edge is the value the XOR statement prescribes to the good. The Hungarian algorithm can be invoked to solve for the maximal weighted matching.

Just like OXS, we can apply the XOR operation to valuations described by OR operations, where each atomic bid is a singleton. This
is called XOR-of-OR-of-Singletons, or XOS for short. Any bid described using OXS can be described using XOS, so the XOS class of bids is more expressive than the OXS class. However, if all bids were in the XOS class, then winner determination is no longer solvable in polynomial time.