We describe several ways to represent bids, and likewise, valuations. We also discuss the complexity of interpreting bids (or valuations).

1 Introduction

In a combinatorial auction environment with $m$ goods, a bid that ascribes a real value to any bundle of goods $G$, $b : 2^G \rightarrow \mathbb{R}$, can require an exponential amount of space to represent, as there are $2^m$ possible bundles. In these notes, we describe several bidding languages that can represent some bids more compactly.

While they are called bidding languages, and can be thought of as languages bidders can use to submit bids to an auctioneer, bidding languages can likewise describe valuations, $v : 2^G \rightarrow \mathbb{R}$. The languages we present are general enough to describe any monotone, normalized bid (or valuation).

**Definition 1.1** (Monotone). A bid (or valuation) is said to be monotone if, for all bundles $S \subseteq G$, the value of any superset $T \supseteq S$ is at least as large as the value of $S$: i.e., $v(S) \leq v(T)$, for all $S \subseteq T \subseteq G$.

**Definition 1.2** (Normalized). A bid (or valuation) is said to be normalized if the value of the empty set is zero: i.e., $v(\emptyset) = 0$.

When an auctioneer asks a bidder how much they value a particular bundle, they are said to be posing a value query. To reply, a bidder must interpret their valuation, meaning evaluate it at the given bundle. These notes describe a variety of bidding constructs; for some of them, interpreting is easy (i.e., in $P$), while for others, it is hard (i.e., $NP$-hard). Interestingly, for one of them (OR-of-XOR) solving a value query for one bidder is equivalent to solving for a welfare-maximizing allocation assuming multiple bidders.

2 The OR/XOR Bidding Language

In these notes, we restrict our attention to two basic operators: XOR and OR. While these are their historical names, their semantics do not coincide with the logical meanings of XOR and OR. XOR is a kind of max operator, and OR, a combination of max and sum.

The syntax of the OR/XOR bidding language is straightforward: we build complex bids by applying XOR and OR operators to atomic
bidding languages 2

bids, much like logical formulae are built by applying logical operators to propositions and relations. Consequently, we first describe the syntax and semantics of an atomic bid; then, we describe the syntax and semantics of the XOR and OR operators.

**Definition 2.1** (Atomic bid). An **atomic bid** is a tuple \((S, p)\), with \(S \subseteq G\) and \(p \in \mathbb{R}_+\).

**Definition 2.2** (Satisfaction). We say that a bundle \(X \subseteq G\) satisfies an atomic bid \((S, p)\) iff \(S \subseteq X\).

We often abbreviate an atomic bid by a Greek letter such as \(\phi\), much as we would represent a logical formula; likewise, we often write \(X \models \phi\) whenever \(X\) satisfies \(\phi\).

An atomic bid \((S, p)\) represents the following valuation: for all \(X \subseteq G\),

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

**Example 2.3.** Imagine an atomic bid \((\{a, b\}, 10)\) that describes a valuation over the set of goods \(\{a, b, c\}\).

- The value of \(\{a, b, c\}\) is 10 because \(\{a, b\} \subseteq \{a, b, c\}\). That is, \(\{a, b, c\} \models (\{a, b\}, 10)\).

- The value of \(\{a, c\}\) is 0 because \(\{a, b\} \not\subseteq \{a, c\}\). That is, \(\{a, c\} \not\models (\{a, b\}, 10)\).

- The value of \(\{c, d\}\) is 0 because \(\{a, b\} \not\subseteq \{c, d\}\). That is, \(\{c, d\} \not\models (\{a, b\}, 10)\).

**Remark 2.4.** Any valuation that can be fully described by a single atomic bid is called a **single-minded** valuation.

Just as logical propositions and relations can be combined using logical operators (e.g., \(\land\), \(\lor\), etc.) to build logical formulae, we can combine atomic bids using logical operators to build complex ones. The first operator we discuss is called XOR, and is written \(\oplus\).

**Definition 2.5** (XOR bid). Given atomic bids \((S_a, p_a)\) for \(1 \leq a \leq A\), an XOR bid takes the form \((S_1, p_1) \oplus (S_2, p_2) \oplus \cdots \oplus (S_A, p_A)\). Such an XOR bid represents the following valuation: for all \(X \subseteq G\),

\[
v(X) = \left( \bigoplus_{a=1}^{A} (S_a, p_a) \right)(X) = \max_{a : X \models (S_a, p_a)} p_a.
\]

In words, the value of a bundle \(X\) according to an XOR bid is the maximal \(p_a\) among all atomic bids \((S_a, p_a)\) that are satisfied by \(X\).
Example 2.6. Imagine an XOR bid \(\{a\}, 4\) \(\oplus\) \((\{b, c\}, 5)\) that describes a valuation over the set of goods \(\{a, b, c\}\).

- The value of \(\{a, b, c\}\) is 5, because \(\{a, b, c\} \models (\{a\}, 4), \{a, b, c\} \models (\{b, c\}, 5)\), and \(\max\{4, 5\} = 5\).
- The value of \(\{a, c\}\) is 4, because \(\{a, c\} \models (\{a\}, 4), \{a, c\} \not\models (\{b, c\}, 5)\), and \(p_{\{a\}} = 4\).
- The value of \(\{c, d\}\) is 0 because \(\{c, d\} \not\models (\{a\}, 4)\) and \(\{c, d\} \not\models (\{b, c\}, 5)\).

Algorithm 1: XOR Interpreter

Input: an XOR bid \(b = \bigoplus_{a=1}^{A} (S_a, p_a)\) and a bundle \(X\)

Output: the value of \(X\) according to \(b\) and the atomic bid that yields that value

1: \(v \leftarrow 0\)
2: \(a \leftarrow \emptyset\)
3: for \(a = 1, \ldots, A\) do
4:   if \(S_a \subseteq X\) then
5:     \(v \leftarrow \max\{v, p_a\}\)
6:     \(a \leftarrow (S_a, p_a)\)
7: end if
8: end for
9: return \(\langle v, a \rangle\)

Computing the value of a bundle according to an XOR bid takes time time linear in the size of the XOR bid (i.e., the number of atomic bids). See Algorithm 1. This is, of course, a good thing. Another good thing is that the XOR bidding language is sufficient to describe any monotone, normalized bid (or valuation). However, expressing an arbitrary bid (or valuation) can require an exponential number of atomic bids. Correspondingly, interpreting an XOR bid can take an exponential amount of time. The OR operator was introduced as a means of potentially reducing the number of atomic bids needed to describe a bid. The OR operator is written \(\lor\).

Definition 2.7 (OR bid). Given atomic bids \((S_a, p_a)\) for \(1 \leq a \leq A\), an OR bid takes the form: \((S_1, p_1) \lor (S_2, p_2) \lor \cdots \lor (S_A, p_A)\).

Interpreting OR bids is not as straightforward as interpreting XOR bids. But it is straightforward to interpret OR bids when they comprise singleton\(^4\) (atomic) bids.

Definition 2.8 (Singleton). A singleton bid is an atomic bid \((S, p)\) s.t. \(S\) is a singleton: i.e., \(|S| = 1\).

Example 2.9. Imagine a singleton bid \((\{a\}, 10)\) that describes a valuation over a set of goods \(\{a, b, c\}\).
• The value of \{a, b, c\} is 10 because \{a, b, c\} \models (\{a\}, 10).
• The value of \{a, c\} is 10 because \{a, c\} \models (\{a\}, 10).
• The value of \{c, d\} is 0 because \{c, d\} \not\models (\{a\}, 10).

Remark 2.10. Any valuation that can be fully described by an XOR bid that comprises singletons is called a **unit-demand** valuation.

The semantics of the OR operator applied to singletons, or any otherwise disjoint atomic bids, is defined as follows: for all \(X \subseteq G\),

\[
v(X) = \left( \bigvee_{a=1}^{A} (S_a, p_a) \right)(X) = \sum_{a : X = (S_a, p_a)} p_a.
\]

In words, the value of a bundle \(X\) according to an OR of disjoint atomic bids is the sum of the \(p_a\)'s of all atomic bids \((S_a, p_a)\) that are satisfied by \(X\). See Algorithm 2.

---

**Algorithm 2: OR of Singletons Interpreter**

**Input:** an OR bid \(b = \bigvee_{a=1}^{A} (S_a, p_a)\) and a bundle \(X\)

**Output:** the value of \(X\) according to \(b\) and the atomic bids that yields that value

1: for all \(g \in X\) do
2: \(v_g \leftarrow 0\)
3: \(a_g \leftarrow \emptyset\)
4: end for
5: for all \(g \in X\) do
6: for \(a = 1, \ldots, A\) do
7: if \(g \in S_a\) then
8: \(v_g \leftarrow \max\{v_g, p_a\}\)
9: \(a_g \leftarrow \{(S_a, p_a)\}\)
10: end if
11: end for
12: end for
13: return \(\sum_{g \in X} v_g, \bigcup_{g \in X} a_g\)

---

**Example 2.11.** Imagine an OR bid \((\{a\}, 4) \lor (\{b\}, 5)\) that describes a valuation over a set of goods \{a, b, c\}.

• The value of \{a, b, c\} is 9, because \{a, b, c\} \models (\{a\}, 4), \{a, b, c\} \models (\{b\}, 5), and 4 + 5 = 9.
• The value of \{a, c\} is 4, because \{a, c\} \models (\{a\}, 4) but \{a, c\} \not\models (\{b\}, 5).
• The value of \{c, d\} is 0 because \{c, d\} \not\models (\{a\}, 4) and \{c, d\} \not\models (\{b\}, 5).
Remark 2.12. A valuation that can be fully described by an OR bid that comprises singletons is called an additive valuation.

Interpreting an OR bid that comprises non-singletons is non-trivial. We defer discussion of this case until after presenting a few more relatively easy cases.

Having understood how to construct complex bids from atomic ones using a single operator (XOR or OR), from a purely syntactic point of view, it is easy to imagine how to construct still more complex bids by combining already complex ones. In particular, we could construct XOR-of-XOR bids, XOR-of-OR bids, OR-of-XOR bids, or OR-of-OR bids. Similarly, we could nest even deeper. The trouble is, some of these bids can be difficult to interpret. Indeed, interpreting OR-of-OR bids is hard, as OR bids alone are hard!

On the other hand, interpreting an XOR-of-OR of singletons—i.e., an XOS bid:

1. Interpret all the (inner) OR clauses using Algorithm 2, which yields a set of atomic bids for each (inner) OR clause.
2. Replace each (inner) OR clause with the corresponding set of atomic bids to create an (outer) XOR bid that comprises only atomic bids.
3. Interpret the (outer) XOR bid using Algorithm 1.

A similar strategy applies to interpreting an OR-of-XOR of singletons (only)—i.e., an OXS bid:

1. Interpret all the (inner) XOR clauses using Algorithm 1, which yields a singleton bid for each (inner) XOR clause.
2. Replace each (inner) XOR clause with the corresponding singleton bid to create an (outer) OR bid that comprises only singleton bids.
3. Interpret the (outer) OR bid using Algorithm 2.

What happens when we try to interpret an OR-of-XOR of non-singleton bids? As we have been suggesting all along, things get more complicated. The reason is, interpreting a single bidder’s OR-of-XOR bid is precisely the same problem as computing a welfare-maximizing allocation, assuming each bidder submits an XOR bid to the auctioneer. That is, it is the NP-hard winner determination problem solved in the inner loop of VCG.

Moreover, interpreting an OR bid (that comprises only atomic bids) is no easier than interpreting an OR-of-XOR bid, because the (inner) XOR clauses can be processed exactly as above, replacing each one with a corresponding atomic bid, yielding an OR bid that
comprises only atomic bids. If this OR bid could be interpreted easily, then WD could be solved easily via this reduction. Hence, it is generally believed that this OR bid cannot be interpreted easily. 

Remark 2.13. Recall that an XOR bid that comprises singletons represents a unit-demand valuation. Interpreting an OR-of-XOR of singletons then is equivalent to solving the winner determination problem assuming all bidders’ valuations are unit-demand. This winner determination problem is in $P$, because it reduces to the problem of finding a maximum weighted bipartite matching.

The semantics of OR bids. An assignment $\alpha$ is a function from a set of goods $G$ to a set of atomic bids $A$: i.e., each good is mapped to at most one atomic bid.\footnote{Goods that unmapped can be understood to map to some dummy bid.} Such an assignment is called satisfying if $\alpha^{-1}(S_a, p_a) \models (S_a, p_a)$, for all atomic bids in $(S_a, p_a) \in A$ s.t. $\alpha^{-1}(S_a, p_a) \neq \emptyset$. Hence, a search for a satisfying assignment is a search for a subset of the atomic bids in $A$ that can all be satisfied simultaneously by the goods in $G$. The value of bundle $X \subseteq G$ according to an OR bid is then the maximal sum, over all satisfying assignments $\alpha$, of the $p_a$’s of the atomic bids $(S_a, p_a)$ that can all be satisfied simultaneously by the goods in $X$.

$$v(X) = \left( \bigvee_{a=1}^{A} (S_a, p_a) \right)(X) = \max_{\alpha : A \rightarrow X} \sum_{a : \alpha^{-1}(S_a, p_a) = (S_a, p_a)} p_a.$$ 

We can operationalize these semantics using a mathematical program, much like the solution to winner determination. We do so here, assuming the set $G$ is made up of entirely distinct goods (no copies of identical goods). In this case, the value of bundle $X \subseteq G$ according to an OR bid is the maximal sum of the $p_a$’s of all non-overlapping atomic bids $(S_a, p_a)$ that can be satisfied by $X$, where two atomic bids $(S_i, p_i)$ and $(S_j, p_j)$ are non-overlapping if $S_i \cap S_j = \emptyset$.

The process of interpreting an OR bid is formulated in terms of these non-overlapping constraints in the following mathematical program:

$$\max_{(T_1, \ldots, T_n) \in \{S_a, p_a\}} \sum_{a : T_a = (S_a, p_a)} p_a$$

subject to $T_i \cap T_j = \emptyset$, \quad $\forall i \neq j \in \{1, \ldots, A\}$

$T_a \subseteq X$, \quad $\forall a \in \{1, \ldots, A\}$

Example 2.14. Imagine an OR bid $(\{a\}, 4) \lor (\{a, b\}, 5)$ that describes a valuation over the set of goods $\{a, b, c\}$.

- The value of $\{a, b, c\}$ is 5, because $\{a, b\} \models (\{a\}, 4)$ and $\{a, b, c\} \models (\{a, b\}, 5)$, but $\{a\} \cap \{a, b\} = \{a\} \neq \emptyset$, so good
a can be assigned to only one of the atomic bids. Value is maximized by assigning it to \((\{a, b\}, 5)\).

- The value of \(\{a, c\}\) is 4, because \(\{a, c\} \models (\{a\}, 4)\) but \(\{a, c\} \not\models (\{a, b\}, 5)\).
- The value of \(\{c, d\}\) is 0 because \(\{c, d\} \not\models (\{a\}, 4)\) and \(\{c, d\} \not\models (\{a, b\}, 5)\).

3 Expressiveness vs. Computational Complexity

XOR bids can, using exponential space, express arbitrary valuations (i.e., any arbitrary function from bundles to values) simply by enumerating all bundles. In spite of this unwieldy space complexity, XOR bids are easy to interpret.

OR bids can express certain valuations, such as additive ones, exponentially more compactly than XOR bids. But OR bids are (believed to be) difficult to interpret.

Regardless, how expressive are OR bids? Can they be used to express an arbitrary valuation? The answer to this question is yes, because any XOR bid can be expressed as an OR bid instead.

To convert an XOR bid to OR bid, we introduce dummy goods. Let \(d\) be a dummy good. Then, given an XOR bid \((S_a, p_a) \oplus (S_b, p_b)\), we can express it as an OR bid as follows: \((S_a \cup \{d\}, p_a) \lor (S_b \cup \{d\}, p_b)\).

Likewise, the XOR bid \((S_a, p_a) \oplus (S_b, p_b) \oplus (S_c, p_c)\) can be expressed as the following OR bid: \((S_a \cup \{d_{ab}\}, p_a) \lor (S_a \cup \{d_{ac}\}, p_a) \lor (S_b \cup \{d_{ab}\}, p_b) \lor (S_b \cup \{d_{bc}\}, p_b) \lor (S_c \cup \{d_{ac}\}, p_c) \lor (S_c \cup \{d_{bc}\}, p_c)\).

**Theorem 3.1.** Let \(A\) be the number of atomic bids in a valuation described by an XOR bid. Such a valuation can be expressed as an OR bid that comprises \(A(A - 1)\) atomic bids using \(1/2 A(A - 1)\) dummy bids.

Although generally exponential in size, the advantage of expressing a valuation as an XOR bid rather than as an OR bid is that interpretation is straightforward. In computer science, there is a well-known tradeoff between expressivity and analyzability: the more expressive a language is, the harder it is to interpret its expressions. With regard to XOR and OR bids specifically, the more compact a language’s expressions are, the more difficult it is to interpret those expressions. Expressive bidding languages that allow us to describe valuations compactly shift the burden from space to time complexity.

---

6 Additive valuations can be expressed using a linear number of atomic bids—one per good—in an OR bid, whereas an exponential number are required in an XOR bid—one per bundle.