TBBL: A Tree-Based Bidding Language for Iterative Combinatorial Exchanges

Ruggiero Cavallo*, **David C. Parkes, Adam I. Juda, Adam Kirsch, Alex Kulesza, Sébastien Lahaie, Benjamin Lubin, Loizos Michael, and Jeffrey Shneidman**
Division of Engineering and Applied Sciences, Harvard University

Abstract

We present a novel tree-based logical bidding language, *TBBL*, for preference elicitation in combinatorial exchanges (CEs). *TBBL* provides new expressiveness for two-sided markets with agents that are both buying and selling goods. Moreover, the rich semantics of *TBBL* allow the language to capture new structure, making it exponentially more concise than OR* and $\mathcal{L}_{GB}$ for preferences that are realistic in important domains for CEs. With simple extensions *TBBL* can subsume these earlier languages. *TBBL* can also explicitly represent partial information about valuations. The language is designed such that the structure in *TBBL* bids can be concisely captured directly in mixed-integer programs for the allocation problem. We illustrate *TBBL* through examples drawn from domains to which it can be (and is being) applied, and motivate further extensions we are currently pursuing.

1 Introduction

The problem of allocating goods efficiently across a set of agents has two central components: determining agents' values over hypothetical allocations (preference elicitation), and using this information to choose an allocation that satisfies certain criteria (winner-determination). In the case of a single-good allocation problem, both of these tasks are relatively simple and many solution methods exist, but the problem becomes significantly more difficult when we seek to allocate multiple goods, where a bidder's value for each individual good may depend on whether or not they receive other goods on the market. Specifically, bidders may consider groups of items *substitutes* or *complements*, leading to sub-additive or super-additive valuation respectively.

A *combinatorial auction (CA)* is a compelling way of approaching allocation problems involving multiple heterogeneous goods. In a CA, bidders are allowed to bid on sets (or bundles) of goods, thus theoretically eliminating the "exposure problem" that results from having to bid on items independently, when in fact the value of an item is dependent on the acquisition of other items. If a bidder were allowed to express a value for every possible bundle up for sale, in principle he could specify his valuation function completely and exactly. However, such an explicit approach is often not feasible given that the number of bundles is exponential in the number of goods. Fortunately, bidders often think about valuations in a structured way, for instance via "business plans", and this structure can be compactly represented through appropriate languages. The trick is to formulate a language that is expressive and concise, but also enables efficient algorithms for winner-determination.

In this paper we present *TBBL*, a tree-based bidding language that, like some previous proposals [Nisan, 2000; Sandholm, 2002; Boutilier and Hoos, 2001, e.g.], provides for expressing values over logical combinations of goods, but has several novel properties. In *TBBL*, valuations are expressed in a tree structure, where internal nodes in the tree correspond to operators for combining subsets of goods, and individual goods are represented at the leaves. The operators on the internal nodes are instantiations of a general schema that allows for arbitrary quantification over children nodes, and defines a particular semantics for propagating value throughout the tree. *TBBL* allows agents to express preferences for both buying and selling goods *in the same tree*. Thus, it is applicable to a *combinatorial exchange (CE)*, a generalization of a CA that is important in many domains. *TBBL* also provides an explicit semantics for *partial* value information: a bidder can specify an upper and lower bound on their true valuation, to be refined during bidding.

TBBL was developed as one crucial piece of a larger, ongoing project: the design and implementation of an iterative CE [Parkes *et al.*, 2005]. A method of converting *TBBL* bids to a mixed-integer program (MIP) formulation of the winner-determination problem has been developed and fully implemented. Design choices for *TBBL* were strongly influenced by our experience thinking about CEs for several real-world domains, including the FAA takeoff and landing-slot allocation problem [Rassenti *et al.*, 1982], the FCC wireless spectrum allocation problem [Kwerel and Williams, 2002], and computational markets such as resource markets on the distributed testbed PlanetLab [Peterson *et al.*, 2002].

2 Background

In a CA there is a set of heterogeneous goods up for sale. A CE is a generalization of a CA in which there are multiple

*Corresponding author. cavallo@eecs.harvard.edu

buyers and sellers, and perhaps agents that both buy and sell. Bidders communicate preferences over possible allocations via *bids*, the syntax and semantics of which is defined by the *bidding language*. Since a bid belongs to an agent, we can consider an allocation to be a mapping from good instances to bids. Bids both define values for allocations and constrain the space of acceptable allocations.

The bidding language plays a key role in both central aspects of the allocation problem, preference elicitation and winner-determination (WD). A bidding language can be evaluated according to the following criteria:

- *expressiveness*: does it, in principle, allow users to fully specify any set of preferences?
- *ease of use*: how hard is it, in practice, for bidders to express their preferences? This relates naturally to the *conciseness* of the language for structured preferences.
- *computational-efficiency*: does the language capture the underlying structure in bidder preferences in a way that allows a computationally tractable winner-determination algorithm to effectively discover the efficient (i.e., value-maximizing) allocation?

It should be noted that the winner-determination problem is equivalent to weighted set-packing, and thus is NP-complete. It tends to be solvable in many practical cases, but care is often required in formulating the problem to capture structure that is present in the domain [Rothkopf *et al.*, 1998; de Vries and Vohra, 2003; Boutilier, 2002].

2.1 Previous logical bidding languages for CAs

Several bidding languages for CAs have previously been proposed, arguably the most compelling of which allow bidders to explicitly represent the logical structure of their valuation over goods via standard logical operators. We refer to these as "logical bidding languages" [Nisan, 2000; Sandholm, 2002; Fujishima *et al.*, 1999]. For instance, an *OR* bid specifies a set of <bundle, price> pairs, where the bidder is willing to buy any number of the specified bundles for their respectively specified prices. This is equivalent to specifying a *set* of single-bundle bids. An XOR bid specifies a set of <bundle, price> pairs, where the bidder is willing to pay for only one of the bundles for its corresponding price. Nisan's OR* language [2000] provides constraints within an OR bid via "phantom variables" (see also [Fujishima *et al.*, 1999]).

One may wonder, given the logical framework adopted by these languages, why they restrict operators to just OR and XOR. The explanation seems to involve characteristics of the accompanying WD-solving methodology the language-designers proposed. Boutilier and Hoos [2000] made the next logical step with the $\mathcal{L}_{GB}$ language, which allows for arbitrarily nested levels combining goods and bundles by the standard propositional logic operators: OR, XOR, and AND. They suggest a further extension that allows bids with a *k-of* operator, used to represent a willingness to pay for any k bundles it quantifies over. In a key insight, Boutilier [2002] specifies a MIP formulation for WD using $\mathcal{L}_{GB}$, and provides positive empirical performance results using a commercial solver, suggesting the computational feasibility of moving to this more expressive logical language.

2.2 What's Missing?

While we believe these previously proposed languages, and particularly $\mathcal{L}_{GB}$, were good steps towards achieving expressive, easy-to-use and efficient representations of bidder preferences, there are still several areas in which they are insufficient. Perhaps most importantly, all of these languages are for the *allocation* of goods, i.e., transfers from a single seller to (potentially) multiple buyers. They do not address the *re-allocation* problem – the task of exchanging goods between agents that may seek to both buy and sell. In other words, none of the previous languages are geared towards CEs, and none would be applicable there without extension. In working in practical domains we also discovered structured valuations that were not concisely represented with existing languages. Finally, the existing languages were not designed with partial-value revelation in mind; this is important in domains where the valuation problem is hard, and can also provide privacy benefits.

3 The *TBBL* Language

TBBL is a logical tree-based bidding language for CEs. It is fully expressive, yet designed to be as concise and structured as possible. It allows for specification of both *bids* and *asks* (a seller's price demand) in a single structure, and allows agents to specify upper and lower bounds on their values for trades. A single *TBBL* bid defines an agent's value (or range of values) for every possible trade. In many cases, it achieves a greater degree of conciseness than previous proposals because of its constraint semantics and general "interval-choose" (IC) class of logical operators, instantiations of which include the standard XOR, OR, and AND. In this section we define the semantics of a *TBBL* bid, and then illustrate these three main novel functional contributions of the language through a series of examples.

3.1 Semantics of a *TBBL* bid

In a *TBBL* bid, individual item trades are represented at the leaves, and internal (non-leaf) nodes correspond to ways of quantifying over lower-level nodes. Every internal node has three properties: an IC operator, a value lower bound, and a value upper bound. The IC operator defines a range of children nodes the bidder is willing to have satisfied. An IC_x^y node (where x and y are non-negative integers) indicates that the bidder is willing to pay for the satisfaction of at least x and at most y of its children. More specifically, *satisfaction* of an IC_x^y node accords to the following rules:

R1 An IC_x^y may be *satisfied* only if at least x and at most y of its children are *satisfied*.

R2 If a node is *not satisfied*, then none of its children may be *satisfied*.

One can consider R1 as a "first pass" that defines a set of candidates for satisfaction, which is then refined by R2. R1 naturally generalizes the approach taken in $\mathcal{L}_{GB}$, where an internal node is satisfied according to its operator and the subset of its children that are satisfied. The semantics of $\mathcal{L}_{GB}$, however, treat logical operators only as a way of specifying when "added value" (positive or negative) results from attain-

ing combinations of goods. In principle, any allocation is "acceptable" to an $\mathcal{L}_{GB}$ bid. We take a different approach. Besides defining how value is propagated, by virtue of R2 our logical operators act as *constraints* on what allocations are *acceptable*. As we demonstrate in Section 3.2, this can provide an exponentially more concise representation in domain-motivated examples. The rule provides a richer semantics.

Leaves of a *TBBL* bid tree are annotated as either *buy* or *sell* nodes; an agent may only specify a sell node for a good that he owns. Typically (i.e., for normal goods), negative value will be declared for *sell* nodes, and positive for *buy* nodes. Intuitively, the declared value indicates the amount the bidder is willing to pay for a trade, so a negative value is a payment demand. Leaves can also be annotated with an (all-or-nothing) quantity.

The *bid-value* of a bid tree is defined relative to an allocation, which constrains the set of leaf nodes that can be satisfied. Every new good assigned to an agent in an allocation will allow an additional buy leaf in that agent's bid tree to be satisfied. Every new good given up by an agent in an allocation requires that an additional sell leaf in that agent's bid tree be considered satisfied. Given an allocation, the bid-value is defined as the sum of the bidder-specified values of all satisfied nodes, where the set of satisfied nodes is chosen to provide the maximal total value.

Formally, for a node β, let v_β denote the value specified at β, and let sat_β equal 1 if the node is satisfied and 0 if not. For a bid tree T and allocation A, the *bid-value* of T given A is defined as follows:

$$v(T, A) = \max_{sat \in valid(T,A)} \sum_{\beta \in T} v_\beta \cdot sat_\beta, \qquad (1)$$

where $valid(T, A)$ defines all ways of choosing the set of satisfied leaf nodes in T consistent with A, and consistent with rules R1 and R2. A mapping to a MIP formulation for *TBBL* is provided in Parkes et al. [2005].

TBBL allows expression of partial value information, with a lower bound $\underline{v}_\beta$ and upper bound $\overline{v}_\beta$ specified for each node β. Lower bounds can be viewed as guarantees on the payment a bidder is willing to make for a given allocation. Upper bounds can be viewed as guarantees on the payment a bidder is *not* willing to make for a given allocation. For instance, Figure 1 portrays the bid tree of a mixed buyer-seller pursuing a swap of good A for good B. The bidder's reported value for A is between 1 and 4, and the value for B is between 2 and 6.

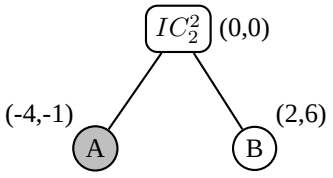


Figure 1: Mixed buyer-seller example with partial value revelation. The shaded node indicates a good the bidder owns.

3.2 Ease-of-Use: Conciseness

Consider the following scenario. An airline has interest in a set of takeoff times at an airport that regulates the number of takeoffs and landings by allocating "time-slots" to different airlines. There are five evening slots (E_1, E_2, E_3, E_4, E_5) and one morning slot (M) on the market, and the airline has a business plan that requires acquisition of the morning slot and 2 or 3 of the evening slots, but no more, where the precise value of the allocation depends on which evening slots are received. The evening slots yield value 1, 2, 3, 4, and 5 respectively to the airline if two or three are acquired along with the morning slot, and there is a bonus of 10 for achieving such a business plan.

Figure 2 portrays the most concise way of representing these preferences in OR*, $\mathcal{L}_{GB}$, and *TBBL*. Both OR* and $\mathcal{L}_{GB}$ must enumerate all combinations of size 2 and 3 of the evening time-slots; *TBBL* need not, because of our specification that no node can be satisfied if its parent is not (R2 above). Note that even if $\mathcal{L}_{GB}$ shared these semantics, the *TBBL* bid would still be significantly more concise due to the increased power of the IC operator over $\mathcal{L}_{GB}$'s *k-of*. The *k-of* operator is equivalent to an IC_k^∞ operator; there is no way of specifying interest in "2 or 3" without applying both a 2-of and a 3-of operator.

3.3 Representing Trades

TBBL is capable of concisely expressing valuations that depend on complex combinations of trades. Consider now a different scenario in which an airline flying out of the slot-controlled airport owns the takeoff rights to three morning time-slots (Tm_1, Tm_2, Tm_3) and two evening slots (Te_1, Te_2), and landing rights to three corresponding morning slots (Lm_1, Lm_2, Lm_3). The airline would naturally want to balance its schedule by acquiring two matching evening landing slots, and may be willing to sell off one morning takeoff-landing pair to do so, but no more than one, and only if it can get the evening slot. If there are three different evening landing slots available, (Le_4, Le_5, Le_6), the airline's valuation function may take the form in Figure 3.

3.4 Partial Value Revelation

Exact value information is only required to determine the efficient allocation in pathological cases in which all allocations are basically tied in value. Typically, to choose between two hypothetical allocations we need only know that the value of one is higher than the other. For instance, Figure 4 portrays an allocation problem with 2 agents, each of which has interest in the other bidder's goods. Bidder 1 will potentially sell one of his items (A or B) if he can get Bidder 2's item, C, at the right price. Bidder 2 is interested in buying either of Bidder 1's goods or selling his own good, with no structural constraints. With the information provided we already know the efficient trade.[1]

The most significant benefits from this partial revelation functionality are brought to bear in *iterative* allocation mech-

[1] A should be transferred to Bidder 2, and C should be transferred to Bidder 1. While we do not know the exact effi ciency yielded from swapping B, the bounds tell us it will not be as great as that from swapping A. While transferring C from Bidder 2 to Bidder 1 may not yield any added value in and of itself, we know it cannot hurt, and since that trade is a prerequisite for Bidder 1 to sell one of his goods, it should be executed.

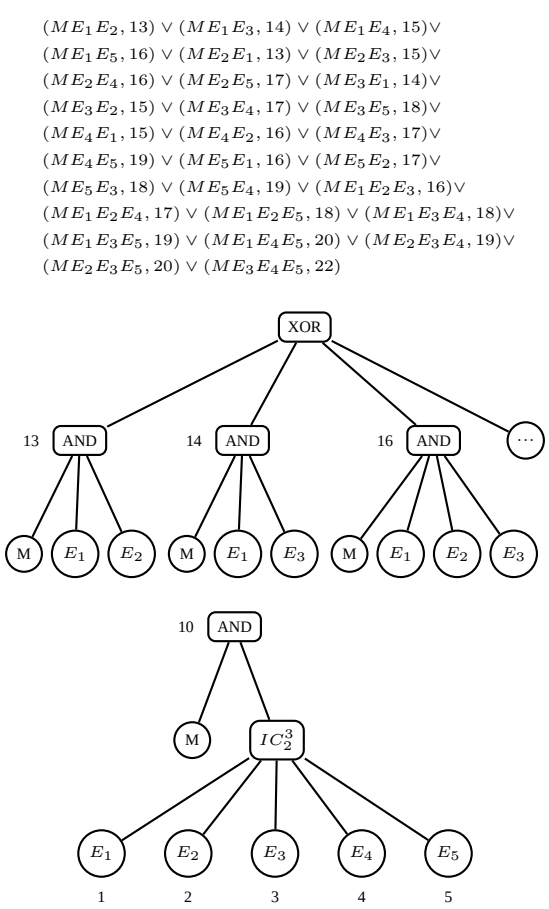


Figure 2: Equivalent representations of the same valuation function in OR*, $\mathcal{L}_{GB}$, and *TBBL* respectively. We have used standard operator names for clarity. Note that XOR = IC_1^1, and AND=IC_x^x where x is the number of children of a node. Both OR* and $\mathcal{L}_{GB}$, but not *TBBL*, have to enumerate all combinations of size 2 and 3 of the evening time-slots. Nodes are annotated with exact value information; where values are omitted, they are assumed to be 0.

anisms, such as Parkes et al.'s Iterative CE (ICE) [2005]. The idea is that bidders can begin with very loose bounds on their valuations, and gradually tighten them in response to pricing information provided by the mechanism. As more information about other agents' values becomes available, bidders are able (and required) to refine their own value range for every node in the tree. Explicit information on partial value information—note that the bidders must bound the uncertainty—is useful in providing feedback and in defining termination conditions in the ICE design.

4 Relationship to Previous Languages

With relatively minor extensions to the semantics described above, *TBBL* subsumes the earlier bidding languages. In this section we describe the relationship of *TBBL* with these extensions to OR* and $\mathcal{L}_{GB}$.

4.1 *TBBL* and OR*

First note that *TBBL* already subsumes the XOR/OR language. While there exist valuations that can be stated more

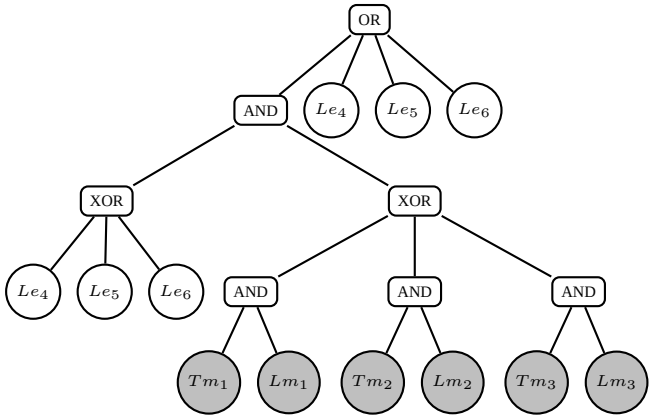
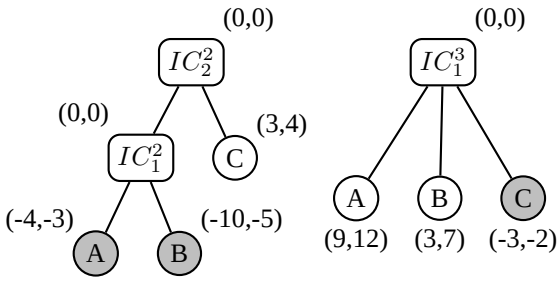


Figure 3: Example demonstrating trading constraints. Note that OR = IC_1^3 in this example. Values are omitted here for simplicity, but all "owned" good nodes (those in gray) would have non-positive value, and all others would have non-negative value. If the bidder values each of its owned morning takeoff-landing pairs equally at v, he could express this by setting those node values to 0, and the XOR that joins them to v.



(a) Bidder 1 (b) Bidder 2

Figure 4: Two bidders, each with partial value information.

concisely in OR* than in *TBBL*, an extension that allows *TBBL* nodes to have multiple parents– thus yielding a *rooted-DAG* (directed acyclic graph) rather than a tree – is more concise than OR*. To see this, note that we can convert any OR* bid to this extended-*TBBL* as follows: construct an OR node, connected to a set of XOR nodes, one for each phantom item in the OR* bid, where each XOR node has children corresponding to bundles in which the respective phantom item appears. For example, see Figure 5 for an OR* bid and its extended-*TBBL* equivalent (g_1 and g_2 are the phantom items).

The number of nodes in the extended-*TBBL* bid (not counting ANDs to unite bundles) will be equal to the number of variables that appear in the OR* bid, including phantoms, plus one (the OR). Since if a phantom item is to be of any use it must appear more than once in an OR* bid, the number of nodes in the extended-*TBBL* bid will be less than or equal to the number of variable specifications in the OR* bid.[2]

[2]Note that *TBBL* without this extension subsumes OR* trivially if we allow phantom items to appear in a *TBBL* bid. We prefer this extension, as it makes the role of phantom items explicit and relates to the multiple-classifi cation problem described in Section 5.1.

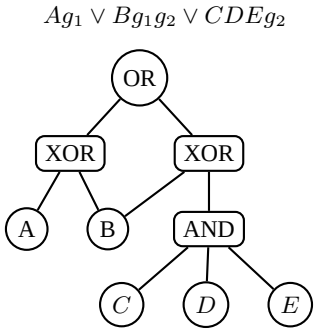
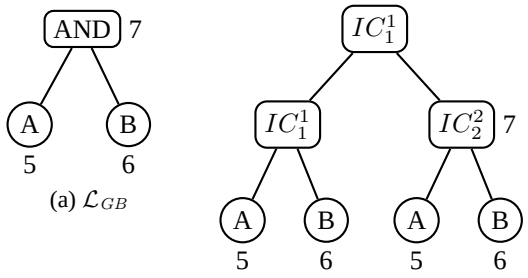


Figure 5: Equivalent OR* and extended-*TBBL* bids.

4.2 *TBBL* and $\mathcal{L}_{GB}$

In Section 3.2 we provided an example demonstrating that *TBBL* can be more concise than $\mathcal{L}_{GB}$. However, the richer semantics that accrue from R2 can also provide less conciseness in some settings. For instance, consider the example in Figure 6, which portrays the most concise representation in each language for a simple two good case, in which $v(A) = 5$, $v(B) = 6$, and $v(AB) = 18$.



(a) $\mathcal{L}_{GB}$

(b) *TBBL* representation

Figure 6: An $\mathcal{L}_{GB}$ bid and its less concise *TBBL* equivalent.

Sometimes the absence of the "downward propagation" of R2 provides opportunities for increased conciseness, but often the rule is essential to avoiding exponential enumeration (e.g., in Figure 2). Thus, we consider a modification of *TBBL* that allows the bidder to specify, for each node, which semantics should be used.

Even more generally, we suggest extending the IC operator to take four parameters, namely two sets of bounds: one that defines the range of children that must be satisfied for *value to propagate*, and one that defines the range for *an allocation to be considered acceptable*. An ${}_a^b IC_x^y$ operator at node β could then be used to express the following: "An allocation is acceptable to me only if between x and y of β's children are satisfied, and I have added value for the allocation at β if between a and b of it's children are satisfied." $\mathcal{L}_{GB}$ is the special case where $x = 0$ and $y = \infty$, and *TBBL* is the special case where $a = x$ and $b = y$.

A further difference between $\mathcal{L}_{GB}$ and *TBBL* is that in $\mathcal{L}_{GB}$ a single item allocation can satisfy multiple leaf nodes. We decided against this approach, but note that this design-issue disappears with the rooted-DAG extension to *TBBL*, in which all leaf nodes that are allowed to be satisfied by a single good are simply represented by a single node that has multiple parents.

5 Missing Semantics & Proposed Constructs

In this section, we consider some extensions to *TBBL* that would facilitate the concise representation of additional structure that we have recognized in thinking about the FAA, FCC and PlanetLab domains. The extensions presented here have not been implemented and future work will need to explore the effect of these features on solver performance.

We draw, in particular, on examples from our experience with PlanetLab, a growing distributed testbed currently composed of about 200 participants worldwide, who each own between two and ten server-class machines that can be simultaneously shared by other users. The resource allocation problem in PlanetLab is to allocate shares of the nearly 550 machines, in various configurations and at various times, to users. We are working on an ICE-based deployment, with *TBBL* as the underlying preference elicitation language.

5.1 Multiple Classification Problem

TBBL is good for preference expression when there is only one *indexing characteristic* (or *attribute*) in the preference statement. An indexing attribute is a property of goods or bundles that is used to mark equivalence and difference, and when combined with value, to indicate preference. In the PlanetLab exchange domain, the primitive good is a machine-timeslot pair. *TBBL* can be used to, e.g., ask for "any two machines for the same timeslot," and the structure of this bid (in the simple case of 3 possible timeslots) is illustrated (ignoring values) in Figure 7.

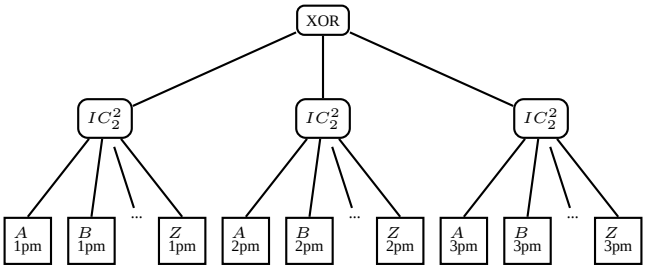


Figure 7: *TBBL* structure for a bid for any two machines during the same timeslot, given three possible timeslots.

In this example, the sole indexing attribute is *time*; however, it is possible that a bid can use more than one attribute for indexing. For instance, consider a user that requires any two machines, each from a different country, running the same operating system, at the same time. Now we have three attributes on which to index. In the current *TBBL* one would express this with leaves for each different attribute-machine combination and then provide internal nodes beneath the XOR to capture the sets of self-consistent attribute groupings. This representation scales exponentially in the number of indexing characteristics. For instance, if there are 50 possible countries and 100 possible timeslots, and each machine can run one of three operating systems, we would expect a *TBBL* bid on the order of 15,000 internal nodes.

Here, we observe that the same rooted-DAG idea from Section 4.2 will reduce the number of internal nodes to (approximately) the sum of indexing characteristic dimensions (153 nodes), rather than the product. A DAG allows one to overlay an independent "structural constraint" tree for each indexing attribute. See Figure 8 for an example. By itself this does not avoid an exponential number of leaves. For this, we are exploring the idea of using "wild characters", e.g., with A** to indicate a good with attribute A and any other two attributes.

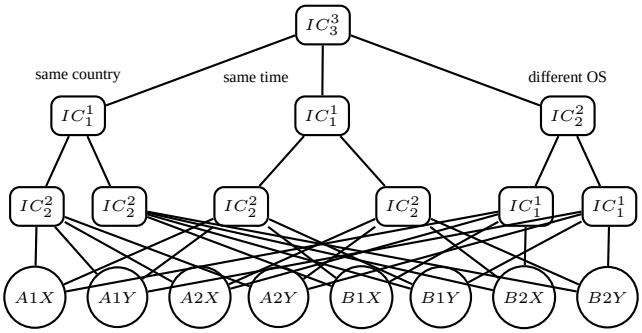


Figure 8: In general, overlaying the indexing characteristic trees yields a graph with far fewer nodes than *TBBL* would require. This picture gives the structure of a bid when there are 2 countries (A and B), two time-slots (1 and 2), and two operating systems (X and Y).

5.2 New Expressiveness: Continuity and Discontinuity

In working in practical domains, we have identified that there is sometimes a need for a more powerful connective than the set-theoretic IC operator. Namely, there are times when a user would like to express relationships between goods that could be more naturally described if the bidder was able to state dependencies between leaf nodes in a bid. As a simple example, a bidder in PlanetLab may wish to bid for "five machines for three time slots, as long as the time slots are consecutive." In the airline domain, an airline might want to express a discontinuity, for instance "three landing slots spaced by at least one hour."

We have been able to show that our MIP formulation for *TBBL* can be readily extended to take into account such continuity operators. In particular, for each node with a continuity operator over its n children, the MIP formulation need only be extended with at most n new linear constraints. In addition, we have strong indications that any reduction of continuity operators to IC operators results in an exponential blowup in the bid size, although we have yet to establish this result formally. Therefore, continuity operators could potentially provide exponentially more concise bids than *TBBL* in certain situations.

Tying continuity operators back to our other proposed extension of *TBBL* to accommodate rooted-DAG bids, we have been able to show that given a rooted-DAG bid, all continuity operators can be reduced to IC operators, with only a polynomial increase in the size of the bid. This provides an indication that the right extension of *TBBL* is towards accommodating rooted-DAG bids, and that continuity operators can simply be provided as syntactic sugar in a user interface on top of the formal language. We are currently investigating these and other related extensions of *TBBL* that would make bids in real-world domains more concise and easier to express.

6 Conclusion

We presented *TBBL*, a new logical bidding language for representing preferences in large, multi-good allocation problems. The most exciting contributions of the language are its added conciseness in general, and its new expressiveness for combinatorial exchanges. We presented several important directions for extension of *TBBL*, motivated by experience with real-world domains. Notably, we have identified significant benefits of moving to a rooted DAG-based representation, and of developing even more expressive logical connectives. We showed that *TBBL* with relatively simple extensions subsumes both OR* and $\mathcal{L}_{GB}$, and is more expressive (in CEs) and more concise than both.

Finally, we note that (like previous languages) *TBBL* was developed to try to compactly represent valuation functions that exist in a space that is exponential in the number of goods; it was not designed to handle an enormous *good space*. In cases where the number of goods is exponential in a number of features or "descriptors," we can potentially find compact representations by moving to *attribute-based* constraint operators. Examples in the PlanetLab domain point to the gains in both efficiency and "ease of use" that may be achieved by such representations.

References

[Boutilier and Hoos, 2001] Craig Boutilier and Holger H. Hoos. Bidding languages for combinatorial auctions. In *Proceedings of the 18th International Joint Conference on Artifi cial Intelligence (IJCAI)*, pages 1211–1217, 2001.

[Boutilier, 2002] C. Boutilier. Solving concisely expressed combinatorial auction problems. In *AAAI*, pages 359–366, 2002.

[de Vries and Vohra, 2003] Sven de Vries and Rakesh V Vohra. Combinatorial auctions: A survey. *Informs Journal on Computing*, 15(3):284–309, 2003.

[Fujishima *et al.*, 1999] Y. Fujishima, K. Leyton-Brown, and Y. Shoham. Taming the computational complexity of combinatorial auctions: Optimal and approximate approaches. In *Proceedings of the 16th International Joint Conference on Artifi cial Intelligence (IJCAI)*, pages 548–553, 1999.

[Hoos and Boutilier, 2000] Holger H. Hoos and Craig Boutilier. Solving combinatorial auctions using stochastic local search. In *AAAI/IAAI*, pages 22–29, 2000.

[Kwerel and Williams, 2002] E. Kwerel and J. Williams. A proposal for a rapid transition to market allocation of spectrum. Technical report, FCC Offi ce of Plans and Policy, 2002.

[Nisan, 2000] Noam Nisan. Bidding and allocation in combinatorial auctions. In *Proceedings of ACM Conference on Electronic Commerce*, pages 1–12, 2000.

[Parkes *et al.*, 2005] David C. Parkes, Ruggiero Cavallo, Nick Elprin, Adam Juda, Sébastien Lahaie, Benjamin Lubin, Loizos Michael, Jeffrey Shneidman, and Hassan Sultan. ICE: An iterative combinatorial exchange. In *Proceedings of ACM Conference on Electronic Commerce*, 2005.

[Peterson *et al.*, 2002] L. Peterson, D. Culler, T. Anderson, and T. Roscoe. A blueprint for introducing disruptive technology into the internet. In *Proceedings of the 1st Workshop on Hot Topics in Networks (HotNets-I)*, 2002.

[Rassenti *et al.*, 1982] Steve J. Rassenti, Vernon L. Smith, and R. L. Bulfi n. A combinatorial mechanism for airport time slot allocation. *Bell Journal of Economics*, 13:402–417, 1982.

[Rothkopf *et al.*, 1998] Michael H. Rothkopf, Aleksandar Pekeč, and Ronald M. Harstad. Computationally manageable combinatorial auctions. *Management Science*, 44(8):1131–1147, 1998.

[Sandholm, 2002] Tuomas Sandholm. Algorithm for optimal winner determination in combinatorial auctions. *Artifi cial Intelligence*, 135(1-2):1–54, 2002.