Modelling Combinatorial Auctions in Linear Logic

Daniele Porello and **Ulle Endriss**
Institute for Logic, Language and Computation
University of Amsterdam

Abstract

We show that linear logic can serve as an expressive framework in which to model a rich variety of combinatorial auction mechanisms. Due to its resource-sensitive nature, linear logic can easily represent bids in combinatorial auctions in which goods may be sold in multiple units, and we show how it naturally generalises several bidding languages familiar from the literature. Moreover, the winner determination problem, i.e., the problem of computing an allocation of goods to bidders producing a certain amount of revenue for the auctioneer, can be modelled as the problem of finding a proof for a particular linear logic sequent.

Introduction

A *combinatorial auction* (CA) is a mechanism for one agent (the *auctioneer*) to sell a set of goods to a number of other agents (the *bidders*). While there are several different types of CAs, in the standard mechanism each bidder first specifies how much they are prepared to pay for any given subset of the set of goods on auction, and the auctioneer then chooses an allocation of goods to bidders that will maximise the sum of payments collected. The advantage of a combinatorial auction over a sequence of simple auctions (one for each individual good) is that it solves the socalled *exposure problem:* in a sequence of simple auctions it would be difficult for a bidder to decide how much to bid for item A, if she is only interested in obtaining A and B *together*; in a CA she can directly express this preference and there is no risk of getting stranded with just A.

While the idea is intuitively appealing, the CA framework also raises a number of challenging research questions: How can we incentivise bidders to truthfully declare their valuations (cf. *game theory*, *mechanism design*)? How can we solve the combinatorial optimisation problem of computing the best allocation given a set of bids (cf. *algorithmics*)? How do we best represent the input of the bidders (cf. *knowledge representation*)? Particularly the first two types of questions have received (and continue to receive) a lot of attention in the literature. The state of the art is reflected in the recent collection edited by Cramton, Shoham, and Steinberg (2006). In this paper, we shall focus on the challenges for knowledge representation raised by CAs.

Knowledge representation techniques play an important role in the design of *bidding languages*. A bid is an encoding of a bidder's (declared) valuation function, which maps *bundles* (subsets of the set of goods on auction) she might receive to the prices she is prepared to pay for them. As the number of conceivable bundles grows exponentially with the number of goods, we require a compact representation language. Several such languages have been proposed, e.g., the socalled XOR- and the OR-languages (Nisan 2006) and languages based on weighted formulas (Boutilier and Hoos 2001). (In fact, as bidding is a form of communicating one's preferences, much of the recent work on *preference modelling* (Goldsmith and Junker 2008) is relevant to this problem.) Importantly, all of the aforementioned bidding languages are languages for *single-unit* CAs, while many real-world auctions are in fact *multi-unit* CAs, where there may be several indistinguishable copies of the same good.

Our first objective will be to overcome this limitation and to design generalisations of standard bidding languages for multi-unit CAs in a systematic and principled way. A natural basis for such an undertaking is the framework of *linear logic* (LL). LL is a resource-sensitive logic: to prove a conclusion from a set of premises, each premise can be used at most once (Girard 1987; 1995). This feature makes it possible to distinguish, for instance, whether a bidder receives one or two copies of the same good, and bidders can quote different prices for these situations.[1]

In fact, we can do much more than that: LL turns out to provide an appropriate framework in which to model a variety of CA mechanisms—and modelling is not restricted to the representation of bids. Our contribution is threefold:

- We show that LL can serve as a basis for designing powerful bidding languages for CAs. Our approach subsumes several existing languages in a single formal framework and adds a number of new features. Specifically, we can model the availability of goods in multiple units and we can distinguish different types of goods, such as goods that are or are not reusable (by the same bidder) or that are or are not sharable (amongst several bidders).
- We show that also the *winner determination problem*

[1] In a somewhat related context (negotiation), similar points have previously been made by Harland and Winikoff (2002) and Küngas and Matskin (2004).

(WDP), i.e., the problem faced by the auctioneer of which goods to award to which bidder so as to maximise revenue, has a natural counterpart in the logical framework. Specifically, we show how to build an "*allocation sequent*" from the goods owned by the auctioneer, the bids received, and the amount of revenue hoped for, and demonstrate that any proof of this sequent corresponds to an allocation yielding the desired revenue. The WDP then reduces to a series of calls to a LL theorem prover.

- Going beyond standard CA mechanisms, we also show how to model more powerful auction mechanisms in LL. This includes, in particular, *mixed auctions* (Cerquides et al. 2007), in which bidders and auctioneer exchange transformations of goods rather than plain goods. Generalising even further, we sketch what we call *formula auctions*, in which the auctioneer sells arbitrary LL formulas to the bidders (roughly speaking, in standard CAs these formulas are conjunctions of atomic propositions, while in mixed auctions they are certain types of implications).

To exemplify our approach, consider the OR-language. An OR-bid is a list of bundles of goods labelled with a price (socalled *atomic bids*): $\langle B_1, w_1\rangle$ OR $\cdots$ OR $\langle B_\ell, w_\ell\rangle$. This bid encodes a valuation function v: a bundle X of goods is said to satisfy a set of atomic bids, if it is a superset of each of the bundles of the atomic bids and if those bundles do not overlap; $v(X)$ is defined as the maximal sum of prices of any set of atomic bids satisfied by X. Later, we will show how to map a multi-unit variant of the OR-language into LL. For example, if bidder 5 wants to express that she will pay one monetary unit (u) for two copies of p and three monetary units for obtaining a copy of p together with a copy of q, then she can submit the following bid (the LL connectives, such as $\otimes$ and $\multimap$, will be introduced in the next section):

$$[(p_5 \otimes p_5) \multimap u] \otimes [(p_5 \otimes q_5) \multimap (u \otimes u \otimes u)]$$

Now, from p_5 we cannot prove anything, from two copies of p_5 we can prove u, from p_5 and q_5 we can prove u^3, from two copies of p_5 and one q_5 we can still only prove u^3, and from three copies of p_5 and one q_5 we can prove u^4.

Let us also briefly sketch our approach for modelling the WDP. Each bid is represented by a formula $\textsc{bid}_i$, like the one shown above. The multiset of goods owned by the auctioneer can be represented by a (multiplicative) conjunction of these goods, e.g., $\textsc{goods} = p \otimes p \otimes q \otimes r \otimes r \otimes r$. We also need a formula that expresses that each of these items can go to (at most) one of the bidders. For example, for (one copy of) p and a group of three bidders, this formula would be $(p \multimap p_1) \,\&\, (p \multimap p_2) \,\&\, (p \multimap p_3)$, using the additive conjunction operator of LL. Let $\textsc{map}$ be the (multiplicative) conjunction of formulas of this kind for each copy of each good. Then the auctioneer can achieve a revenue of k if and only if there exists a proof for the following sequent:

$$\textsc{goods}, \textsc{map}, \textsc{bid}_1, \ldots, \textsc{bid}_n \vdash u^k$$

Moreover, the allocation achieving that level of revenue can be read off the proof. Solving the WDP then amounts to finding the largest value k such that the above sequent can be proved, and then extracting the corresponding allocation from that proof. We stress that we do not intend to propose LL as an *algorithmic* framework for solving the WDP. This will continue to require highly specialised combinatorial optimisation algorithms. Instead, we view this embedding as an attractive conceptual framework in which to model and understand a wide variety of different CA mechanisms and bidding languages in a principled manner.

The remainder of this paper is organised as follows. After recalling the basic concepts of LL, we first show how to embed (multi-unit variants of) three important bidding languages into LL. We then show how to model the problem of finding a suitable allocation as the problem of finding a proof for a LL sequent of the kind outlined above and we provide a formal proof of this correspondence. Before concluding, we discuss a number of possible extensions of the basic framework, including mixed auctions and general formula auctions.

Background on Linear Logic

In this section, we review the relevant notions from LL. For full details, the reader is referred to Girard (1995) and Troelstra (1992). LL provides a resource-sensitive account of proofs by means of a controlled use of the structural rules of weakening and contraction within the sequent calculus:

$$\frac{\Gamma \vdash \Delta}{\Gamma, A \vdash \Delta}\,\mathrm{W} \qquad \frac{\Gamma, A, A \vdash \Delta}{\Gamma, A \vdash \Delta}\,\mathrm{C}$$

Removing the structural rules, we are lead to split the usual connectives into two classes, since, for example, the following presentations of rules are not equivalent anymore:

$$\frac{\Gamma \vdash A \qquad \Gamma' \vdash B}{\Gamma, \Gamma' \vdash A \wedge B}\,\mathrm{R}\wedge \qquad \frac{\Gamma \vdash A \qquad \Gamma \vdash B}{\Gamma \vdash A \wedge B}\,\mathrm{R}\wedge$$

Without structural rules, sequents behave as multisets of formula occurrences and we have to distinguish connectives that take the concatenation of contexts (*multiplicatives*) and connectives that demand a shared context (*additives*).

Given a set of positive atoms $\mathcal{A}$, the language of LL is defined as follows (where $p \in \mathcal{A}$): $\quad L \;::=$

$$p \mid \mathbf{1} \mid \bot \mid \top \mid \mathbf{0} \mid L^\perp \mid L \otimes L \mid L \parr L \mid L \oplus L \mid L \& L \mid\, !L \mid\, ?L$$

Linear negation $(\cdot)^\perp$ is involutive and each formula in LL can be transformed into an equivalent formula where negation occurs only at the atomic level. The conjunction $A \otimes B$ ("tensor") means that we have exactly one copy of A and one copy of B, no more no less. Thus, e.g., $A \otimes B \nvdash A$. We might say that in order to sell A and B, we need someone who buys A and B, while here there is just a buyer for A. We will not directly use the disjunction $A \parr B$ ("par"); rather we use linear implication: $A \multimap B := A^\perp \parr B$. Linear implication can be seen as a form of deal: "for A, I sell you B". The additive conjunction $A \,\&\, B$ ("with") introduces a form of choice: we have one of A and B and we can choose which one. For example, $A \,\&\, B \vdash A$, but we do not have them both: $A \,\&\, B \nvdash A \otimes B$. The additive disjunction $A \oplus B$ ("plus") means that we have one of A and B, but we cannot choose, e.g., $A \vdash A \oplus B$ but $A \oplus B \nvdash A \,\&\, B$. The exponentials $!A$ and $?A$ reintroduce structural rules in a local way: !-formulas licence (C) and (W) on the lefthand

$$\frac{}{A \vdash A}\,\text{ax} \qquad \frac{\Gamma, A \vdash C \qquad \Gamma' \vdash A}{\Gamma, \Gamma' \vdash C}\,\text{cut}$$

MULTIPLICATIVES

$$\frac{\Gamma, A, B \vdash C}{\Gamma, A \otimes B \vdash C}\,\otimes\text{L} \qquad \frac{\Gamma \vdash A \qquad \Gamma' \vdash B}{\Gamma, \Gamma' \vdash A \otimes B}\,\otimes\text{R}$$

$$\frac{\Gamma \vdash A \qquad \Gamma', B \vdash C}{\Gamma', \Gamma, A \multimap B \vdash C}\,\multimap\text{L} \qquad \frac{\Gamma, A \vdash B}{\Gamma \vdash A \multimap B}\,\multimap\text{R}$$

$$\frac{\Gamma \vdash C}{\Gamma, \mathbf{1} \vdash C}\,\mathbf{1}\text{L} \qquad \frac{}{\vdash \mathbf{1}}\,\mathbf{1}\text{R}$$

ADDITIVES

$$\frac{\Gamma, A_i \vdash C}{\Gamma, A_0 \& A_1 \vdash C}\,\&\text{L} \qquad \frac{\Gamma \vdash A \qquad \Gamma \vdash B}{\Gamma \vdash A \& B}\,\&\text{R}$$

$$\frac{\Gamma, A \vdash C \qquad \Gamma, B \vdash C}{\Gamma, A \oplus B \vdash C}\,\oplus\text{L} \qquad \frac{\Gamma \vdash A_i}{\Gamma \vdash A_0 \oplus A_1}\,\oplus\text{R}$$

$$\frac{}{\Gamma, \mathbf{0} \vdash C}\,\mathbf{0}\text{L} \qquad \frac{}{\Gamma \vdash \top}\,\top\text{R}$$

EXPONENTIALS

$$\frac{\Gamma, A \vdash C}{\Gamma, !A \vdash C}\,!\text{L} \qquad \frac{!\Gamma \vdash A}{!\Gamma \vdash !A}\,!\text{R}$$

STRUCTURAL RULES

$$\frac{\Gamma, A, B, \Gamma' \vdash C}{\Gamma, B, A, \Gamma' \vdash C}\,\text{P} \qquad \frac{\Gamma, !A, !A, \vdash C}{\Gamma, !A \vdash C}\,!\text{C} \qquad \frac{\Gamma \vdash \Delta}{\Gamma, !A \vdash \Delta}\,!\text{W}$$

Table 1: Sequent Calculus for Intuitionistic LL

side of $\vdash$; ?-formulas licence (C) and (W) on the right. Intuitively, exponential formulas can be copied and erased; they are relieved from their linear status.

We will use the intuitionistic version of linear logic (ILL), obtained by restricting the righthand side of the sequent to a single formula; so for example we will not have $?$ and $⅋$ in the language. In fact, we will mostly use ILL augmented with the global weakening rule (W). The reasons for these choices will become clear later. The rules of the sequent calculus for ILL are shown in Table 1 (Troelstra 1992).

To control complexity, we can restrict attention to certain fragments: *intuitionistic multiplicative linear logic* (IMLL) using only $\otimes$ and $\multimap$; *intuitionistic multiplicative additive linear logic* (IMALL) using only $\otimes$, $\multimap$, $\&$ and $\oplus$; and *Horn linear logic* (HLL). In the latter, sequents must be of the form $X, \Gamma \vdash Y$ (Kanovich 1994), where X and Y are tensors of positive atoms, and Γ is one of the following (with X_i, Y_i being tensors of positive atoms):

(i) Horn implications: $(X_1 \multimap Y_1) \otimes \cdots \otimes (X_n \multimap Y_n)$

(ii) &-Horn implications: $(X_1 \multimap Y_1) \& \cdots \& (X_n \multimap Y_n)$

For these fragments we can rely on the following proof-search complexity results. MLL is NP-complete and so is MLL with full weakening (W) (Lincoln 1995). The same results apply for the intuitionistic versions. HLL is NP-complete, and so is HLL + W (Kanovich 1994). MALL and IMALL are PSPACE-complete (Lincoln et al. 1992).

Bidding Languages

In this section, we provide three examples for bidding languages that can be represented using different types of Horn fragments of LL. These are the well-known and widely used XOR- and OR-languages (Nisan 2006), as well as the language of k-additive valuations (Chevaleyre et al. 2008), which itself is an instance of the framework of bidding languages based on weighted propositional formulas (Boutilier and Hoos 2001; Uckelman et al. 2009). Importantly, while all the works just cited discuss bidding languages for auctions in which goods are available in *single units*, in what follows we shall present languages that are also suitable for *multi-unit* CAs.

In a multi-unit CA, an auctioneer wants to sell the elements of a finite multiset of goods $\mathcal{M}$ (with finite multiplicity) to a group of bidders. Let $\mathcal{M}(p)$ denote the multiplicity of item p in $\mathcal{M}$. We define the set of *atoms* $\mathcal{A} = \{p_1, \ldots, p_m\}$ as the set of elements of $\mathcal{M}$ ignoring their multiplicity.

There is an isomorphism between multisets and tensor formulas of atoms (up to associativity and commutativity):

$$\{m_1, \ldots, m_k\} \;\cong\; m_1 \otimes \cdots \otimes m_k$$

Thus, we can represent each subset $X \subseteq M$ as a tensor product. Moreover, if $M \cong A$ and $N \cong B$, then the (disjoint) union of M and N is isomorphic to $A \otimes B$.

We now want to define languages to encode *valuations* $v : \mathcal{P}(\mathcal{M}) \to \mathbb{N}$, mapping subsets of $\mathcal{M}$ to prices.[2]

Atomic Bids

To model prices symbolically, we assume a finite set of distinct weight atoms $\mathcal{W} = \{w_1, ..., w_p\}$. In fact, often we will use just one weight atom u. We write u^k for the tensor product $u \otimes \cdots \otimes u$ (k times). To associate weights with numbers, we define a function $val : \mathcal{W} \to \mathbb{N}$, with $val(u) = 1$. Let $\mathcal{W}^{\otimes}$ be the set of all finite tensor products of atoms in $\mathcal{W}$, modulo commutativity (including the "empty" product $\mathbf{1}$). That is, $\mathcal{W}^{\otimes} = \{\mathbf{1}, w_1, w_2, w_1 \otimes w_2, \ldots\}$. We extend val to $\mathcal{W}^{\otimes}$ by stipulating $val(\mathbf{1}) = 0$ and $val(\varphi \otimes \psi) = val(\varphi) + val(\psi)$. In particular, this means that $val(u^k) = k$.

Definition 1. *An atomic bid is a formula of the form $B \multimap w$, where B is a tensor product of atoms in $\mathcal{A}$ and $w \in \mathcal{W}$.*

In a CA, given a bid $B \multimap w$, we can work with two alternative assumptions: *no free disposal* at the bidder's side, meaning that the bidder will pay w if she receives *exactly* B, and *free disposal* at the bidder's side, meaning that the bid is satisfied whenever the bidder receives *at least* B. In the sequel, unless otherwise stated, we will always assume

[2] For ease of notation, we shall assume $0 \in \mathbb{N}$.

free disposal. To model free disposal, we will use ILL with weakening (W).[3]

Definition 2. *Every bid formula* BID *generates a valuation* v_{BID} *mapping multisets* $X \subseteq \mathcal{M}$ *to prices:*

$$v_{\text{BID}}(X) = \max\{val(w') \mid w' \in \mathcal{W}^{\otimes} \text{ and } X, \text{BID} \vdash w'\}$$

Definition 2 applies to atomic bids as well as to the more powerful bidding languages we will define in the sequel. In the case of atomic bids $\text{BID} = (B \multimap w)$, it simply says that $v_{B\multimap w}(X) = w$ whenever X is equal to a superset of the multiset isomorphic to B, and $v_{B\multimap w}(X) = 0$ otherwise.

In case the only weight atom used is u, i.e., if $\mathcal{W} = \{u\}$, then Definition 2 can be simplified and we obtain:[4]

$$v_{\text{BID}}(X) = \max\{k \mid X, \text{BID} \vdash u^k\}$$

XOR-bids

An XOR-bid $\langle B_1, w_1\rangle$ XOR $\cdots$ XOR $\langle B_\ell, w_\ell\rangle$ expresses that a bidder would like to get at most one of the bundles she specifies, for the associated price (Nisan 2006). In LL, this idea can be captured via the additive conjunction ($\&$).

Definition 3. *An XOR-bid is a formula of the form*

$$(B_1 \multimap w_1) \;\&\; \dots \;\&\; (B_\ell \multimap w_\ell),$$

where each B_i *is a tensor product of atoms in* $\mathcal{A}$ *and each* w_i *is a weight atom from* $\mathcal{W}$.

Definition 2 provides the semantics for XOR-bids by fixing the valuation functions they generate.

Example 4. *Given an XOR-bid* $(p \multimap u) \;\&\; (q \multimap w) \;\&\; (p \otimes q \otimes r \multimap z)$*, suppose the auctioneer provides* $\{p, p, q, r, s\}$*. Using these goods, it is possible to satisfy each of the atomic bids in the XOR-bid. For example, the auctioneer can satisfy the bid producing* z*:*

$$\dfrac{\dfrac{p, q, r, p \otimes q \otimes r \multimap z \vdash z}{p, p, q, r, s, p \otimes q \otimes r \multimap z \vdash z}\,W}{p, p, q, r, s, (p \multimap u) \;\&\; (q \multimap w) \;\&\; (p \otimes q \otimes r \multimap z) \vdash z}\,\&L$$

However, we have to choose which atomic bid to satisfy, according to the meaning of $\&$.

Example 5. *We define two classes of valuation functions, adapting their definitions from Nisan (2006) to the multi-unit case. The simple additive valuation,* $v(X) = |X|$ *for* $X \subseteq \mathcal{M}$*, can be expressed via the following formula, which is exponential in size in the number of items in* $\mathcal{M}$ *(we slightly abuse the notation identifying the multiset* B *with the corresponding tensor formula):*

$$\&_{B \subseteq \mathcal{M}} (B \multimap u^{|B|})$$

The simple unit demand valuation, $v(X) = 1$ *for* $X \neq \emptyset$ *and* $v(\emptyset) = 0$*, can be expressed in the XOR-language via:*

$$(p_1 \multimap u) \;\&\; \cdots \;\&\; (p_m \multimap u)$$

[3] Alternatively, we could use the additive constant of linear logic $\top$ and write bids $B \otimes \top \multimap w$ to make it explicit in the syntax that a bidder has free disposal.

[4] We can define $u^0 = \mathbf{1}$. Using weakening (to represent free disposal), from $\vdash \mathbf{1}$ we get $\Gamma \vdash \mathbf{1}$, for any Γ. So every bid produces u^0, since it will always be satisfied by any allocation (also by allocating nothing), e.g., $p, p \otimes q \multimap u^k \vdash \mathbf{1}$ will be provable.

We say that a valuation $v : \mathcal{P}(M) \to \mathbb{N}$ is *monotonic* if and only if for all $X_1, X_2 \subseteq \mathcal{M}$, if $X_1 \subseteq X_2$, then $v(X_1) \leq v(X_2)$. Recall that we can model both free disposal or the lack thereof simply by using $\vdash$ with and without weakening (W), respectively. Following Nisan (2006) and Cerquides et al. (2007) we can easily prove that, also in our framework, the XOR-language without free disposal can express all valuations and the XOR-language with free disposal is fully expressive over the space of monotonic valuations.

Proposition 6. *The following hold:*

(1) *Every valuation* $v : \mathcal{P}(\mathcal{M}) \to \mathbb{N}$ *is generated by some XOR-bid without free disposal.*

(2) *XOR-bids with free disposal generate all monotonic valuations and only those.*

Proof. (1) Given a function v: $\mathcal{P}(\mathcal{M}) \to \mathbb{N}$, for each pair $(X, h) \in v$, define an atomic bid $(x_1 \otimes \cdots \otimes x_\ell \multimap \bar{h})$ where $x_1 \otimes \cdots \otimes x_\ell \cong X$ and $\bar{h}$ is a weight symbol for h. Joining all the atomic bids via $\&$, we have a complex bid $\&_i \text{BID}_i$ generating the function $v_{\&_i \text{BID}_i}$. Now, for any $Y \subseteq \mathcal{M}$ we get $v_{\&_i \text{BID}_i}(Y) = v(Y)$, since *the only* w' we can prove with the sequent $Y, \&_i \text{BID}_i \vdash w'$ is the weight associated with Y.

(2) In one direction, if a function v is generated by an XOR-bid BID with free disposal, then, given $X_1 \subseteq X_2$, if $X_1, \text{BID} \vdash w'$, by applying weakening, we also have $X_2, \text{BID} \vdash w'$. Hence, $\{w' \mid X_1, \text{BID} \vdash w'\} \subseteq \{w' \mid X_2, \text{BID} \vdash w'\}$ and therefore $v(X_1) \leq v(X_2)$. For the other direction, we can take the construction in the proof of part (1), but now allowing for weakening. $\square$

OR-bids

An OR-bid $\langle B_1, w_1\rangle$ OR $\cdots$ OR $\langle B_\ell, w_\ell\rangle$ states that a bidder agrees to receive any number of disjoint bundles at the sum of their prices (Nisan 2006). The appropriate LL connective for modelling this kind of semantics is the tensor ($\otimes$).

Definition 7. *An OR-bid is a formula of the form*

$$(B_1 \multimap w_1) \otimes \dots \otimes (B_\ell \multimap w_\ell),$$

where each B_i *is a tensor product of atoms in* $\mathcal{A}$ *and each* w_i *is a weight atom from* $\mathcal{W}$.

The intended meaning of a tensor/OR-bid is that the bidder would pay the sum of the corresponding w_i for each bundle of goods B_i she gets. The formal semantics of OR-bids is again given by Definition 2.

The usual condition on OR-bids, namely that the required bundles of goods do not overlap, works well if goods are available in single units: since we are here considering the multi-unit case, the condition of not being allowed to overlap is replaced by imposing that the right amount of goods is provided in order to satisfy the atomic bids in the OR-bid. For example, the OR-bid $\langle p, 1\rangle$ OR $\langle p, 1\rangle$ will be fully satisfied only if the auctioneer provides two copies of p. This is the meaning of the provability of a sequent containing OR-bids in Definition 2.

Example 8. *Given an OR-bid $(p \otimes q \multimap v) \otimes (q \multimap w)$, suppose the auctioneer provides $\{p, q\}$. The OR-bid can be satisfied in two possible ways:*

$$\dfrac{\dfrac{p, q, p \otimes q \multimap v \vdash v}{p, q, p \otimes q \multimap v, (q \multimap w) \vdash v}\,W}{p, q, (p \otimes q \multimap v) \otimes (q \multimap w) \vdash v}\,\otimes L$$

or:

$$\dfrac{\dfrac{\dfrac{q, q \multimap w \vdash w}{p, q, q \multimap w \vdash w}\,W}{p, q, (p \otimes q \multimap v), q \multimap w \vdash w}\,W}{p, q, (p \otimes q \multimap v) \otimes (q \multimap w) \vdash w}\,\otimes L$$

The definition of the valuation generated by OR-bids then lets us take the maximum of w and v.

Example 9. *In the OR-language we can express the simple additive valuation by means of the following formula:*

$$\bigotimes_{i \in \{1,\dots,m\}} [\underbrace{(p_i \multimap u) \otimes \cdots \otimes (p_i \multimap u)}_{\mathcal{M}(p_i)\ \textit{times}}]$$

Observe that the OR-language is only attractive if we do assume free disposal (i.e., weakening); without it, it has the same expressive power as the simple language of atomic bids. For example, without free disposal, $(p \multimap u^k) \otimes (q \multimap u^{k'})$ and $p \otimes q \multimap u^{k+k'}$ generate the same valuation.

It is interesting to remark that the usual characterisation of the expressivity of the OR-language for single-unit CAs (Nisan 2006) cannot straightforwardly be extended to the multi-unit case. In the single-unit case, OR-expressions generate functions v such that $v(X \cup Y) \geq v(X) + v(Y)$, whenever $X \cap Y = \emptyset$. If we try to apply the same condition to the multi-unit case, taking the disjoint union of X and Y, we do not arrive at a correct characterisation of the expressivity of the OR-language. Take the expression $\textsc{or} : (p \multimap u) \otimes (p \otimes p \multimap u)$. We have that the generated function will provide a value of 1 on $\{p, p\}$, $v_{\textsc{or}}(\{p, p\}) = 1$, which is less than $v(\{p\}) + v(\{p\}) = 2$. The problem is connected with the interpretation of the marginal value that can be associated with various copies of a same item. Moreover, since we are dealing with multisets of finite multiplicity, the valuations generated by our languages cannot grow arbitrarily, so at a certain point the function generated by the OR-expression will provide a constant value.

We leave the full investigation of the expressivity of our tensor language to future work.

K-additive Languages

The language of k-additive valuations (Chevaleyre et al. 2008) is based on the idea of specifying weights for the *marginal* valuations derived from sets of goods, rather than directly specifying the values of full bundles. Let $\mathcal{M}[k]$ be the set of all multisets $Y \subseteq \mathcal{M}$ such that $|Y| \leq k$. A valuation v is called *k-additive* if there exists a mapping $v' : \mathcal{M}[k] \to \mathbb{Z}$ such that $v(X) = \sum\{v'(Y) \mid Y \subseteq X \text{ and } Y \in \mathcal{M}[k]\}$. The notion of k-additivity gives rise to a bidding language: by specifying a (marginal, possibly negative) price for each bundle of size $\leq k$ (as an atomic bid) we can represent v' and thus v.

The class of k-additive languages are a special case of the family of languages based on weighted propositional formulas (Uckelman et al. 2009). Such languages have been widely studied in the AI literature; for the specific use in CAs they have first been proposed by Boutilier and Hoos (2001). A *goalbase* G is a set of pairs (φ, w), where φ is a proposition (in classical logic) and w is a weight. G induces a valuation that maps any assignment of truth values to atoms to the sum of the weights of the formulas that are satisfied by that assignment (which we can think of as a bundle of goods). A characterisation of k-additive valuations in logical terms is provided by Uckelman et al. (2009); the class of k-additive functions is proved to be equivalent to the class of functions generated by goalbases of *positive cubes*, i.e., conjunctions of positive literals, $(p_1 \wedge \cdots \wedge p_\ell, w)$.

A difference between the OR-language and goalbase languages (including k-additive languages) is that the accepted atomic bids may overlap. For example, in $G = \{(p \wedge q, 5), (p, 3)\}$, the allocation of p and q will satisfy both atomic bids. In our framework, this means that the allocated goods are not consumed within a goalbase. We define atomic bids that interpret goods as being *reusable* as formulas of the form $(B_i \multimap B_i \otimes w_i)$, where B_i is a tensor of atoms.

Definition 10. *A k-additive bid is a formula of the form*

$$(B_1 \multimap B_1 \otimes w_1) \otimes \cdots \otimes (B_\ell \multimap B_\ell \otimes w_\ell),$$

where each B_i is a tensor product of atoms in $\mathcal{A}$ and each w_i is a weight atom from $\mathcal{W}$.

The semantics of k-additive bids is given by Definition 2. Note that we can also mix different kinds of bids, e.g., bids that do and do not consume goods (OR- and k-additive bids). We will discuss in more detail the relationship between different types of resources later.

Example 11. *Suppose $G = \{(p \otimes q \multimap p \otimes q \otimes v), (p \multimap p \otimes w)\}$. If the auctioneer provides p and q, then all the atomic bids in G are satisfied:*

$$\dfrac{p \vdash p \qquad \dfrac{\dfrac{p, q \vdash p \otimes q \qquad \dfrac{p, q, v, w \vdash v \otimes w}{p \otimes q \otimes v, w \vdash v \otimes w}\,\otimes L}{p, q, p \otimes q \multimap p \otimes q \otimes v, w \vdash v \otimes w}\,\multimap L}{p \otimes w, p \otimes q \multimap p \otimes q \otimes v \vdash v \otimes w}\,\otimes L}{p, q, p \multimap p \otimes w, p \otimes q \multimap p \otimes q \otimes v \vdash v \otimes w}\,\otimes R$$

Regarding the expressivity of k-additive bids, it is possible to adapt the relevant results of Uckelman et al. (2009) to the case of multiple units and to our LL framework.

Remark 12. *Intuitionistic (and classical) logic can be translated into LL (Girard 1995). Define the translation $(\cdot)^*$ as follows: $p^* = p$, $(A \wedge B)^* = A^* \,\&\, B^*$, $A \to B = !(A^*) \multimap B^*$, $(A \vee B)^* = A^* \oplus B^*$. We have that: $\Gamma \vdash_{IL} A$ if and only if $!\Gamma^* \vdash_{LL} A^*$. So we can translate any goalbase into a LL formula with the same logical behaviour, in the sense that they will be satisfied by the same sets of resources. However, the full power of exponentials makes LL with weakening, though decidable (Kopylov 1995), exponential-space hard (Urquhart 2000), while full LL is undecidable (Lincoln et al. 1992). Thus, while in principle one can model the interaction of bounded and unbounded resources (sets and multisets) in LL, the price to pay is complexity.*

The Allocation Problem

In this section, we formulate the problem of computing an allocation producing a certain amount of revenue as the problem of finding a proof for a LL sequent. This allows us, at least in principle, to model the winner determination problem as a series of calls to a LL theorem prover.

Let $\mathcal{M}$ again be a multiset of goods owned by the auctioneer, and let $\mathcal{N} = \{1, ..., n\}$ be the set of bidders. We add to the set of atoms $\mathcal{A} = \{p_1, \ldots, p_m\}$ all atoms p_i^j to express that the good p_i is allocated to the individual j. From now on, we will assume that bids are defined using these indexed names of goods, i.e., bidder $j \in \mathcal{N}$ must express her bid using the set of atoms $\{p_1^j, \ldots, p_m^j\}$.

In order to express that each (copy of) a good may be allocated to any of the bidders (but not to more than one), we shall use the following formula:[5]

$$\text{MAP} \ := \bigotimes_{p \in \mathcal{A}} [\&_{j \in \mathcal{N}} (p \multimap p^j)]^{\mathcal{M}(p)} \qquad (1)$$

Given bids $\text{BID}_1, \ldots, \text{BID}_n$, an *allocation* yielding revenue k is a function $\alpha : \mathcal{M} \to \mathcal{N} \cup \{*\}$ with $\sum_i v_{\text{BID}_i}(A_i) = k$, where $A_i = \alpha^{-1}(i)$ and $\alpha^{-1}(*)$ are the unallocated goods.

We now define the concept of *allocation sequent*, which is intended to capture the problem, faced by the auctioneer, of finding a feasible allocation returning a particular revenue. We restrict ourselves to the case of $\mathcal{W} = \{u\}$. We take $\mathcal{M}$ and $\mathcal{N}$ to be fixed, and MAP to be defined accordingly.

Definition 13. *The allocation sequent for revenue k and bids $\text{BID}_1, \ldots, \text{BID}_n$ is defined as the following LL sequent:*

$$\mathcal{M}, \text{MAP}, \text{BID}_1, \ldots, \text{BID}_n \vdash u^k$$

We are now ready to state the relationship between proofs and actual allocations.

Proposition 14. *Given n bids in any of the bidding languages introduced (XOR, OR, k-additive), every allocation α with revenue k provides a proof π of an allocation sequent for k, and vice versa, every proof π of an allocation sequent for k provides an allocation α with revenue k.*

Proof. We sketch the main steps of the proof.

($\Rightarrow$) Let $\alpha : \mathcal{M} \to \mathcal{N} \cup \{*\}$ be an allocation for $\text{BID}_1, \ldots, \text{BID}_n$ yielding revenue k. W.l.o.g. assume the first $l \leq n$ are the bidders receiving a nonempty bundle. Let $A_1, \ldots, A_l$ be those nonempty subsets of $\mathcal{M}$, i.e., $k = \sum_{j \leq l} v_{\text{BID}_j}(A_j)$. For each $j \leq l$, we define A_j^j as the multiset of atoms in A_j indexed with the name of bidder j. By definition, if $v_{\text{BID}_j}(A_j^j) = val(w)$, then $A_j^j, \text{BID}_j \vdash w$. So we can start building the proof π, applying ($\otimes$R):

$$\frac{A_1^1, \text{BID}_1 \vdash w_1 \qquad \ldots \qquad A_l^l, \text{BID}_l \vdash w_l}{\underbrace{a_1^1, \ldots, a_{h_1}^1}_{A_1^1}, ..., \underbrace{a_1^l, \ldots, a_{h_l}^l}_{A_l^l}, \text{BID}_1, \ldots, \text{BID}_l \vdash w_1 \otimes \cdots \otimes w_l}$$

For each $a_j^j \in A_1^1 \cup \cdots \cup A_l^l$, we use axioms $a_j \vdash a_j$; so we get by application of ($\multimap$L):

$$\frac{\boxed{a_j \vdash a_j} \qquad a_1^1, \ldots, a_{h_1}^1, \ldots, a_1^l, \ldots, a_{h_l}^l, \text{BID}_1, \ldots, \text{BID}_l \vdash w_j}{a_1^1, \ldots, \boxed{a_j, a_j \multimap a_j^j}, \ldots, a_{h_l}^l, \text{BID}_1, \ldots, \text{BID}_l \vdash w_j}$$

From each $a_j \multimap a_j^j$ we can build MAP, inferring, by $n-1$ applications of (&L), the formula $\&_{j \in \mathcal{N}}(a_j \multimap a_j^j)$.

If $A_1, \ldots, A_l$ equals the full multiset of goods $\mathcal{M}$, then we are done. Otherwise, we can weaken the proof by introducing atoms in $\alpha^{-1}(*)$ and formulas $(c \multimap c^1) \& \cdots \& (c \multimap c^n)$, for each $c \in \alpha^{-1}(*)$.

($\Leftarrow$) Given any proof π of an allocation sequent, we can transform it as follows. First, we can move the application of weakening down. Then we can also delay the application of & in such a way that every application of (&R) is below any application of ($\otimes$R).[6] So we obtain a proof π' such that there is no application of weakening and (&L) above the step:

$$\frac{\pi'}{\mathcal{M}', a \multimap a_j, b_1, \ldots, b_q \vdash u^k} \qquad (2)$$

where $\mathcal{M}' \subseteq \mathcal{M}$, $a \multimap a_j$ are some of the &-conjuncts composing MAP, and each b_i may be an atomic bid, a part of a k-additive bid, a part of an OR-bid, or an atom in an XOR-bid. Since π' is proved without weakening and &, π' is provable in MLL. Sequents in MLL are *balanced* (Lincoln et al. 1992): the number of positive and negative atoms occurring in the sequent must be the same. So, using step (2), we can define $A_j = \{a_j \mid a_j \in \pi'\}$, since those are the goods actually used to satisfy bids in π'. □

In this way, we could import known algorithms for winner determination for CAs into our framework. On the other hand, given a proof π in the fragments we saw, we can transform it into a cut-free proof in polynomial time (Girard, Scedrov, and Scott 1992). In a cut-free proof, each connective is visited exactly once, so given a proof of the allocation sequent, we can retrieve an allocation in polynomial time.

For the three languages presented, allocation sequents belong to HLL, so the complexity of checking whether revenue k is attainable is in NP (Kanovich 1994), meaning that our form of modelling the problem does not increase complexity with respect to the standard approach (Cramton, Shoham, and Steinberg 2006). Of course, Proposition 14 only provides a method for solving the *decision variant* of the WDP. In practice, we will want to find the maximal revenue k such that u^k is provable. This can be achieved by using binary search over possible values of k and checking the corresponding allocation sequents in turn.

Extensions

Next, we discuss several extensions of our basic framework for modelling CAs in LL. We shall restrict ourselves to brief examples illustrating the main ideas.

[5]Formula (1) is required in order for our approach to work with k-additive languages, since here we have to model that, on the one hand, goods are *reusable* within the bid of a single bidder and, on the other, goods are not *sharable* across the bids of distinct bidders. If we were to restrict attention to XOR- and OR-languages, then we could do without indexed goods and without formula (1).

[6]Permutation rules in LL have been fully investigated by Galmiche and Perrier (1994).

Enriching the Language

In LL, we can distinguish between *sharable* goods, between bidders, *reusable* goods, for one bidder, and simple consumable goods. The idea that LL may be useful in designing bidding languages that can distinguish sharable from non-sharable goods has already been hinted at by Boutilier and Hoos (2001). We can define a bounded form of exponential as $!^\ell\varphi$, meaning that we can use φ at most ℓ times (Girard, Scedrov, and Scott 1992). We can then define the full availability of a good for the bidders as $!^\ell p$ where ℓ is big enough, so $!^\ell p$ can be shared by all bidders demanding it. In order to express the reusability of a good for a single bidder j, we can write $!^\ell p^j$, which will satisfy only bidder j's bids. In order to make explicit that j can reuse p as much as she likes, we can add the formula $p^j \multimap !^\ell p^j$ to j's bid formula.

Mixed Auctions

In *mixed auctions* (Cerquides et al. 2007), bids are encoding valuations over multisets of *transformations*. A transformation is an input-output pair (I, O) of multisets of goods, indicating to the auctioneer that the bidder is willing to produce O if supplied with I (as well as to pay an associated price). An atomic bid $\langle (I, O), w\rangle$ will be satisfied by the allocation of transformation (I', O') if I' is enough to satisfy the bidder's demand I ($I' \supseteq I$) and O is enough to satisfy the auctioneer's demand O' ($O' \subseteq O$). In LL, we can define mixed atomic bids as $I \multimap O \otimes w$. We can define bidding languages on top of atomic bids as before; and the valuation generated by a complex mixed bid BID is given by:

$$v_{\text{BID}}(I, O) = \max\{val(w') \mid w' \in \mathcal{W}^{\otimes} \text{ and } I, \text{BID} \vdash O \otimes w'\}$$

Example 15. *Suppose the auctioneer's available input I is $\{p, q\}$ and she wants to obtain an output O of (at least) $\{r, s\}$, using the following transformations offered by the bidders: $p \multimap r \otimes t \otimes v$, $t \multimap s \otimes p \otimes u$. Since the sequent is provable, there is a feasible allocation:*

$$[q, p]_I, p \multimap r \otimes t \otimes v, t \multimap s \otimes p \otimes u \vdash [r \otimes s]_O \otimes p \otimes u \otimes v$$

As in the work of Cerquides et al. (2007), the XOR-language can be proved to be fully expressive with respect to valuations defined over transformations. Moreover, an allocation of transformations to bidders in the sense of Cerquides et al. (2007) provides a proof π of the sequent

$$I, \text{BID}_1, \ldots, \text{BID}_n \vdash O \otimes u^k,$$

and, *vice versa*, given a proof π we can define an allocation as we did in the previous section.

Formula Auctions

We could in principle generalise the languages we saw allowing for *any* kind of formula to be a bid. Generalising even further, we could replace $\mathcal{M}$ (a tensor of atoms) with an arbitrary formula A. This leads to what we call a *formula auction*, in which, intuitively speaking, the auctioneer owns a "big" formula A, the bidders submit their bids $\text{BID}_1, \ldots, \text{BID}_n$ (arbitrary formulas, using an indexed alphabet), and the WDP amounts to finding "small" formulas $B_1, \ldots, B_n$ such that $A \vdash B_1 \otimes \ldots \otimes B_n$ and $B_j^j, \text{BID}_j \vdash w_j$ (where B_j^j is the indexed version of B_j) for all bidders j and the sum of the values of the weight atoms w_j is maximal. (If desired, this can also be combined with the specification of a required output for the auctioneer.)

Interestingly, our approach also extends to this very general form of one-to-many negotiation. Indeed, it is possible to construct a *single* sequent that corresponds to the WDP:

$$A, \text{MAP}, \text{BID}_1, \ldots, \text{BID}_n \vdash C[u^k] \tag{3}$$

Here, C is the output[7] which may contain a tensor formula u^k representing payments; and MAP is a generalisation of formula (1) that can be defined by induction on formulas. The provability of sequent (3) entails the feasibility of the exchange (the demands match the supplies). The complexity bounds of the proof search for sequent (3) will depend exclusively on the language in which formulas are defined.

Conclusion

We have argued that linear logic provides a powerful formal framework in which to model combinatorial auctions. Not only does LL allow us to extend several of the standard bidding languages to the multi-unit case in a generic manner, but we can also model the procedural aspects of auctions inside the logical framework, by relating the winner determination problem of auctions to the notion of provability.

Future work will include (1) the use of *proof nets* (Girard 1995) to simplify the structure of proofs and to provide a semantic (functional) interpretation of the auction itself, and (2) modelling objective functions other than sum-taking so as to extend the approach from combinatorial auctions (with utilitarian aggregation) to other forms of resource allocation (e.g., fair division with egalitarian aggregation).

Acknowledgements. We would like to thank the reviewers of this paper for their helpful feedback.

References

Boutilier, C., and Hoos, H. H. 2001. Bidding languages for combinatorial auctions. In *Proc. 17th International Joint Conference on Artificial Intelligence (IJCAI-2001)*.

Cerquides, J.; Endriss, U.; Giovannucci, A.; and Rodríguez-Aguilar, J. A. 2007. Bidding languages and winner determination for mixed multi-unit combinatorial auctions. In *Proc. 20th International Joint Conference on Artificial Intelligence (IJCAI-2007)*.

Chevaleyre, Y.; Endriss, U.; Estivie, S.; and Maudet, N. 2008. Multiagent resource allocation in k-additive domains: Preference representation and complexity. *Annals of Operations Research* 163(1):49–62.

Cramton, P.; Shoham, Y.; and Steinberg, R., eds. 2006. *Combinatorial Auctions*. MIT Press.

Galmiche, D., and Perrier, G. 1994. On proof normalization in linear logic. *Theor. Comput. Sci.* 135(1):67–110.

[7] We have used intuitionistic sequents, because they have a *single* output. Thus, we can always see which formula in the sequent provides the revenue.

Girard, J.-Y.; Scedrov, A.; and Scott, P. J. 1992. Bounded linear logic: a modular approach to polynomial-time computability. *Theor. Comput. Sci.* 97(1):1–66.

Girard, J.-Y. 1987. Linear logic. *Theor. Comput. Sci.* 50(1):1–101.

Girard, J.-Y. 1995. Linear logic: Its syntax and semantics. In *Advances in Linear Logic*. Cambridge University Press.

Goldsmith, J., and Junker, U. 2008. Preference handling for artificial intelligence (editorial). *AI Magazine* 29(4):9–12.

Harland, J., and Winikoff, M. 2002. Agent negotiation as proof search in linear logic. In *Proc. 1st International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2002)*.

Kanovich, M. I. 1994. The complexity of Horn fragments of linear logic. *Annals of Pure and Applied Logic* 69(2-3):195–241.

Kopylov, A. P. 1995. Decidability of linear affine logic. In *Proc. 10th Annual IEEE Symposium on Logic in Computer Science (LICS-1995)*. IEEE Computer Society.

Küngas, P., and Matskin, M. 2004. Symbolic negotiation with linear logic. In *Proc. 4th International Workshop on Computational Logic in Multiagent Systems (CLIMA IV)*. Springer-Verlag.

Lincoln, P.; Mitchell, J. C.; Scedrov, A.; and Shankar, N. 1992. Decision problems for propositional linear logic. *Annals of Pure and Applied Logic* 56(1–3):239–311.

Lincoln, P. 1995. Deciding provability of linear logic formulas. In *Proc. Workshop on Advances in Linear Logic*. Cambridge University Press.

Nisan, N. 2006. Bidding languages for combinatorial auctions. In *Combinatorial Auctions*. MIT Press.

Troelstra, A. S. 1992. *Lectures on Linear Logic*. CSLI Publications.

Uckelman, J.; Chevaleyre, Y.; Endriss, U.; and Lang, J. 2009. Representing utility functions via weighted goals. *Mathematical Logic Quarterly* 55(4):341–361.

Urquhart, A. 2000. The complexity of linear logic with weakening. In Buss; Hájek; and Pudlák., eds., *Logic Colloquium '98*, Lecture Notes in Logic. AK Peters.