EE 382V Social Computing **Fall 2018**

Lecture 21: October 27

Lecturer: Vijay Garg *Scribe: Robert Pate*

21.1 Background on Lattice Linear Predicates

We proved some things about stable marriage, such as stable matching always exists, but with Lattice Linear Predicates we can prove it more easily and apply the same approach to the biggest problems such as housing allocation, market clearing, etc.

We had algorithms already for all of these problems but this gives us extra insight into them.

21.2 Consistent Global State

Definition 21.1 *Lattice*

A poset where a join and meet exist for all pairs of elements

Definition 21.2 *Distributive Lattice*

Lattice where joins and meets distribute over each other.

For example we have an example poset below where there are 4 elements, the events are proposals, and the orderings indicate the order the proposals were done.

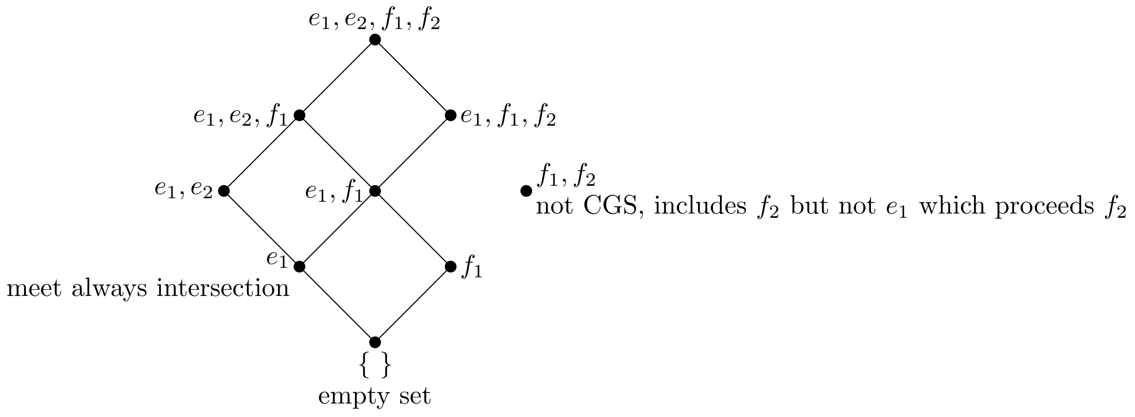
$$\begin{tikzcd} e_1 \arrow[r] \arrow[dr] & e_2 \\ f_1 \arrow[r] & f_2 \end{tikzcd}$$

From this poset we define a 2nd poset $X(e_1, e_2, f_1, f_2)$ which has the same 4 elements. There are 16 possible subsets, but we don't want all of them. We want just the "consistent global states." This is also used in distributive systems by treating this as the order of events in a system.

Definition 21.3 *Consistent Global State*

G, a subset of X, is a CGS if
$\forall\ e, f \in X\ :\ (f \in G) \wedge (e < f) \Rightarrow e \in G$

Graph of X drawn upward:



We started with the smallest at the bottom which is always the empty set, and draw each of the consistent global states. This also draws a natural ordering relationship between them because our $<$ relation is defined by being a subset of.

Claim 21.4 *A set of CGS is a distributive lattice*

Proof: *Definitions:*
Elements: all are CGS due to our claim
Lattice: our relation must be poset ("subset of" is reflexive, anti-symmetric, and transitive)
Distributive: must be closed under meets and joins (a meet of multiple CGS is an intersection of them)

So we must show:
G is a CGS $\wedge$ H is a CGS $\Rightarrow G \cap H$ is a CGS

Meet is greatest lower bound. $G \cap H$ is lower than G because any intersection is smaller. It's also the greatest lower bound because we can have elements lower than $G \cap H$.

1. Pick any two events where:
$e, f \in X$
$f \in G \cap H$
$e < f$
Now we prove $e \in G \cap H$:

2. $f \in G \cap H$ implies $(f \in G) \wedge (f \in H)$.

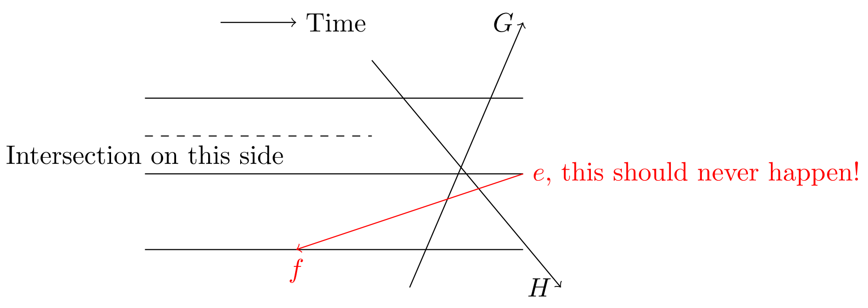
3. $(e < f) \wedge (G$ is CGS$) \Rightarrow e \in G$.

4. $(f \in H) \wedge (e < f) \Rightarrow e \in H$

5. $(e \in H) \wedge (e \in G) \Rightarrow e \in G \cap H$

■

21.2.1 Drawing a poset on a time line



There cannot be arrows going backwards (such as e, f as pictured above) in a CGS since it would no longer be consistent. Consistent means for every included event, all prior events are also included. In the above picture, if we have $e < f$ but e is not in G nor H, they would not be consistent.

21.2.2 Unions

We've shown this works for meets, a similar argument can be made to show it's also closed under joins where the relation is unions instead of intersections.

21.3 Lattice Linear Predicates

Back to our example, these events could be proposals from men to women or proposals from buyers to sellers. And the subsets tells us what has happened so far and tells us the state of the system.

It depends on the problem you're looking at, but in the example of the Stable Matching Problem, a state is "good" if the current state of proposals forms a stable matching.

Our goal would be to find the states that are good for us. We'll define a boolean predicate that's much easier to analyze called a Lattice Linear Predicate.

Definition 21.5 *Lattice Linear Predicate*

Given a distributive lattice L and given a boolean predicate B, B is a Lattice Linear Predicate if:
$\forall$ *CGSs G:* $\neg B(G) \Rightarrow \exists$ *i:forbidden*(G, i, B)

LLP Example

If you're happy, you're done.

If not, there must be some process, such as a man in SMP, that if it doesn't move then you cannot reach a good state (such as a stable marriage). If we can find him, then we can advance him to proceed toward stable marriage.

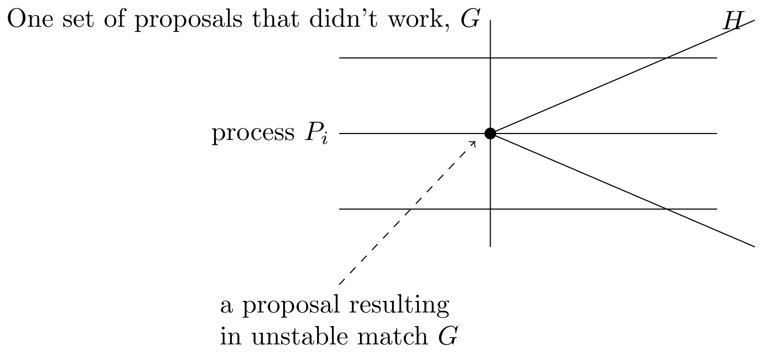
Back to the lattice, start with the lowest set (always empty set) then proceed to e_1 or f_1. The problem is that the lattice could be huge! If some predicate is LLP it saves us from having to explore the entire lattice. It lets us know a given combination is bad so we can throw it away and continue. Any point that we find is forbidden we know all older points are also forbidden since we're dealing with Consistent Global States. For the same reason, and any future states where we haven't advanced the problem process are also forbidden.

Definition 21.6 *Berkoff's Theorem*

In any CGS, if my predicate is not true, unless I advance, it can never be true.

Definition 21.7 *Forbidden*

Given any distributive lattice L and predicate B
forbidden $(G, i, B) \equiv$
$\forall H \in L : G \leq H :$
$(G[i] = H[i]) \Rightarrow \neg B(H)$



Consider the point pictured above. Think of it as a proposal that results in an unstable matching G. There are at least one of these such that if we continue around it we'll never succeed. In any state in the future from here (such as H) it will not be good because of this blocking pair.

This is valuable in helping us prune our search space to speed up our algorithms, if we can find it.

Simple Example Forbidden Process:

$$\begin{tikzcd}
p_1 & e_1 \arrow[r] \arrow[dr] & e_2 \\
p_2 & f_1 \arrow[r] & f_2
\end{tikzcd}$$

Given a predicate $B(G) \equiv$ "any state which contains at least one event from p_2." Unless we execute one event from p_2, we'll never make the predicate true. So p_2 is the forbidden process, and $\{\}, \{e_1\}, \{e_1, e_2\}$ are all bad states.

Claim 21.8 *B is a lattice linear if and only if B is "closed under meets."*
B is true in $G \wedge B$ is true in $H \Rightarrow B(G \cap H)$.

If true in 2 places, find meet. If false then not lattice linear. So if not true in e_1, should be at least 1 dimension that if I don't progress on it I can never make it true. So if I can advance something else to make B true then e_1 is not forbidden.

21.4 Lattice Linear Algorithm

21.4.1 Recap

1. Defined notion of predicate on a lattice

2. Defined which predicates are good
3. Defined closure of meets

Some predicates are nice from both meet and join perspectives, such as SMP, but not all. Payoff: if algorithm is lattice linear then it's efficient

21.4.2 LLP Algorithm to find the minimum feasible vector

- One variable: where are you in the lattice, as a global vector G
- init G as the least CGS in the lattice
- return G as the least CGS that is feasible, e.g. minimum market clearing price
- B feasible predicate: an assignment is feasible if for example it's a stable marriage or market clearing price
- Set an upper bound T that is the top most element, such as the number of women. Top is the end, bottom is the start.
- Note this is a generic algorithm, but "forbidden" will differ for each type of problem
- "Successor" is defined in respect to chain in poset. Go to next event on the same process.
- Never define the whole lattice, it's huge. break poset into chains (however you want) and call them processes, so long as lattice linear.
- Max time: total count of events $O(n^2)$

```
 vector function getLeastFeasible ($T$: vector, $B$: predicate)
   var $G$: the least CGS in the lattice
   while ($\exists$ $j$ : forbidden $(G, j, B)$ do
        for all $j$ in forbidden$(G, j, B)$:
             if $(G[j] = T[j])$ then return null // reached the top, no feasible vector
             else $G[j]$ := successor of $G_j$
end while
return $G$
```

$\forall j$: forbidden (G, j, B) is equivalent to $\neg B(G)$ because B is lattice linear. If this is not true it implies there must exist something that is forbidden. So if your predicate is not true, find one that is forbidden and advance that.

21.5 Proving lattice-linearity of predicates

Lemma 21.9 *Lattice Linearity has the property:*
if B_1 *is LL* $\wedge$ B_2 *is LL* $\Rightarrow$ $B_1 \wedge B_2$ *is LL*

Proof:

1. Pick any global state and if it's not true here, we can find a forbidden

2. Find the forbidden: if $B_1 \wedge B_2$ is not true, then either B_1 is not true or B_2 is not true.
3. Say B_1 is not true, and we know B_1 is LL then there exists at least one place we can advance to make it true
4. You'll find at least one process on which you must advance

■

21.6 Conclusion

Goal for next session is be able to show:

- SMP predicate is lattice linear
- Market clearing price is lattice linear
- Housing allocation problem is lattice linear
- And that the same algorithm works for all 3