# On Some Tractable Cases of Logical Filtering

**T. K. Satish Kumar** and **Stuart Russell**
Computer Science Division
University of California, Berkeley
{tksk, russell}@eecs.berkeley.edu

**Abstract**

*Filtering* denotes any method whereby an agent updates its *belief state*—its knowledge of the state of the world—from a sequence of actions and observations. In *logical filtering*, the belief state is a logical formula describing the possible world states. Efficient algorithms for logical filtering bear important implications on reasoning tasks such as planning and diagnosis. In this paper, we will identify classes of transition constraints that are amenable to compact and indefinite filtering—presenting efficient algorithms wherever necessary. We will first show that connected row-convex (CRC) constraints are amenable to efficient filtering when path-consistency is enforced in appropriate steps. We will then extend this theory to provide a filtering algorithm based on repeatedly enforcing path-consistency and embedding the domain values of the related variables in tree structures to guarantee global consistency. Finally, we will identify and comment on the problem of multi-agent localization as a potential application of the theory developed in the paper (under some reasonable assumptions).

## Introduction

When an agent operates in a partially observable environment, it must maintain a representation of its knowledge about the world. Filtering denotes any method whereby an agent updates its belief state—its knowledge of the state of the world—from a sequence of actions and observations. In stochastic models, for example, the *Kalman filter* (Kalman 1960) maintains a multivariate Gaussian belief state over $n$ system variables, assuming linear Gaussian transition and observation models. In each step of the Kalman filter, the cost of updating the belief state is $O(n^3)$, and the space requirement for maintaining the belief state is $O(n^2)$. Since these costs do not depend on the length of the observation sequence, a Kalman filter can run indefinitely. In logical domains, however, the belief state is best represented as a logical formula describing the possible world states; and efficient *logical filtering* refers to the task of having to maintain a compact representation of the belief state even when we have to deal with a potentially unbounded sequence of actions and observations.

In the most general version of the logical filtering problem, the *initial state* may be only partially known; the *transition model* (which allows for actions by the agent) may be nondeterministic; and the *observation model* may be nondeterministic and partial—i.e., the agent may not be able to observe the actual state. Filtering is closely related to the computational problems arising in many important contexts: including planning, diagnosis, game playing, etc. In planning, for example, maintaining a compact representation of the reachability information (as in a planning graph) is known to be a crucial factor in the success of many recent planners—whether or not they deal with nondeterminism in the actions and/or the initial state (for examples, see (Nguyen and Kambhampati 2000), (Bryce and Kambhampati 2005) and (Cushing and Bryce 2005)). Very similar issues are also addressed in filtering when nondeterminism is allowed in the initial state, transition model, and the observation model; and in general, any tractable cases of the filtering problem would bear important implications on our ability to efficiently deal with situations where we are required to maintain a compact representation of the belief state (see (Amir and Russell 2003)).

The computational costs associated with a filtering algorithm include: (1) the time needed to update the belief state, and (2) the space required to represent it. These complexities depend on: (a) the nature of the uncertainty in the initial state, (b) the nature of the transition model (which describes how the system evolves over time), (c) the nature of the observation model (which describes the way in which the environment generates observations), and (d) the family of representations used to represent the belief state (see (Amir and Russell 2003)). It is well known that even when we restrict ourselves to propositional logic, the general filtering problem is hard; the hardness caused mainly because of the need to represent an exponentially large number of possible world states.

In this paper, we will deal with the filtering problem by abstracting it into a temporally extended constraint satisfaction problem (CSP); in particular, we will identify classes of transition constraints (and observation models) that are amenable to compact and indefinite filtering—presenting efficient algorithms wherever necessary. We will first show that CRC constraints are amenable to efficient filtering when path-consistency is enforced in appropriate steps. We will

```
ALGORITHM: PATH-CONSISTENCY
INPUT: A binary constraint network $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$.
OUTPUT: A path-consistent network.
   (1) Repeat until no constraint is changed:
      (a) For $k = 1, 2 \dots N$:
         (i) For $i, j = 1, 2 \dots N$:
            (A) $R_{ij} = R_{ij} \cap \Pi_{ij}(R_{ik} \bowtie D_k \bowtie R_{kj})$.
END ALGORITHM
```

Figure 1: Shows the basic algorithm for enforcing path-consistency in a binary constraint network. Here, $\Pi$ indicates the projection operation, and $\bowtie$ indicates the join operation (similar to that in database theory).

$$\begin{array}{c|cc} X_1 \backslash X_2 & d_{21} & d_{22} \\ \hline d_{11} & 0 & 0 \\ d_{12} & 1 & 0 \\ d_{13} & 0 & 1 \end{array} \qquad \begin{array}{c|ccc} X_2 \backslash X_3 & d_{31} & d_{32} & d_{33} \\ \hline d_{21} & 0 & 1 & 1 \\ d_{22} & 0 & 1 & 0 \end{array}$$

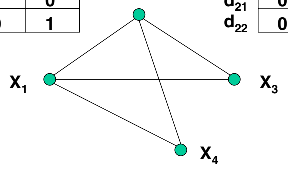
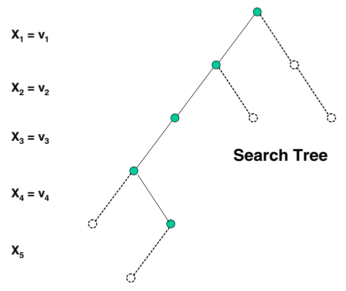


Figure 2: Shows an example of a CSP where the domains of the variables are ordered, and the binary constraints are represented as (0,1)-matrices. The ordered domains of the variables $X_1$, $X_2$ and $X_3$ are respectively $\langle d_{11}, d_{12}, d_{13} \rangle$, $\langle d_{21}, d_{22} \rangle$ and $\langle d_{31}, d_{32}, d_{33} \rangle$ respectively (shown in the figure). For clarity, only two of the constraints are shown in their matrix representations.

then extend this theory to provide a filtering algorithm based on repeatedly enforcing path-consistency and embedding the domain values of the related variables in tree structures to guarantee global consistency. (In turn, we will comment on generalizing this theory to perform filtering via an iterative enforcement of increasing levels of local consistency followed by appropriate geometric embeddings of the domain values of the related variables to guarantee global consistency.) Finally, we will identify and comment on the problem of multi-agent localization as a potential application of the theory developed in the paper (under some reasonable assumptions).

## Preliminaries and Definitions

A CSP is defined by a triplet $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$, where $\mathcal{X} = \{X_1, X_2 \dots X_N\}$ is a set of variables, and $\mathcal{C} = \{C_1, C_2 \dots C_M\}$ is a set of constraints on subsets of them. Each variable $X_i$ is associated with a discrete-valued domain $D_i \in \mathcal{D}$, and each constraint $C_i$ is a pair $\langle S_i, R_i \rangle$ defined on a subset of variables $S_i \subseteq \mathcal{X}$, called the *scope* of $C_i$. $R_i \subseteq D_{S_i}$ ($D_{S_i} = \times_{j \in S_i} D_j$) denotes all compatible tuples of $D_{S_i}$ allowed by the constraint. A *solution* to a CSP is an assignment of values to all the variables from their respective domains such that all the constraints are satisfied.



Ordered Domain of $X_5$

| | $d_{51}$ | $d_{52}$ | $d_{53}$ | $d_{54}$ | $d_{55}$ |
|---|---|---|---|---|---|
| $X_1 = v_1$ | ✓ | ✓ | ✓ | | |
| $X_2 = v_2$ | | | ✓ | ✓ | ✓ |
| $X_3 = v_3$ | | ✓ | ✓ | ✓ | |
| $X_4 = v_4$ | ✓ | ✓ | | | |

Figure 3: The left side of the figure shows a partial search tree associated with solving a CSP. The figure illustrates the instant of time when we have successfully instantiated a few of the variables (say $X_1$ to $v_1$, $X_2$ to $v_2$, $X_3$ to $v_3$ and $X_4$ to $v_4$), and we are searching for a consistent extension to the next variable (say $X_5$). The right side of the figure shows the domain elements of $X_5$ that are consistent with the values assigned to the previously instantiated variables ($X_1 = v_1$, $X_2 = v_2$, $X_3 = v_3$ and $X_4 = v_4$); tick marks indicate consistent combinations, and blanks indicate inconsistent combinations. We consider the case when the tick marks appear consecutively in each row (assuming an ordering on the domain values of $X_5$).

A network of binary constraints is *arc-consistent* if and only if for all variables $X_i$ and $X_j$, and for every instantiation of $X_i$, there exists an instantiation of $X_j$ such that $R_{ij}$ is satisfied. Similarly, a network of binary constraints is *path-consistent* if and only if for all variables $X_i$, $X_j$ and $X_k$, and for every instantiation of $X_i$ and $X_j$ that satisfies the direct relation $R_{ij}$, there exists an instantiation of $X_k$ such that $R_{ik}$ and $R_{kj}$ are also satisfied. Conceptually, algorithms that enforce path-consistency work by iteratively "tightening" the binary constraints as shown in Figure 1.

The best known algorithm that implements the procedure in Figure 1 exploiting low-level consistency maintenance is presented in (Mohr and Henderson 1986), and has a time complexity of $O(N^3K^3)$ (where $K$ is the size of the largest domain). This algorithm is optimal, since even verifying path-consistency has the same lower bound. When binary relations are represented as matrices, path-consistency algorithms employ the three basic operations of *composition*, *intersection* and *transposition*. The (0,1)-matrix representation of a relation $R_{ij}$ (denoted $M_{R_{ij}}$) between variables $X_i$ and $X_j$ consists of $|D_i|$ rows and $|D_j|$ columns when orderings on the domains of $X_i$ and $X_j$ are imposed. The '1's and '0's in the matrix respectively indicate the allowed and disallowed tuples.[1] Figure 2 presents an example of a CSP with matrix notations for the constraints.

## CRC Constraints

A binary relation $R_{ij}$ represented as a (0,1)-matrix, is *row-convex* if and only if, in each row, all of the '1's are consecutive. It has been shown in (Van Beek and Dechter

[1]An extension of this representation mechanism to non-binary constraints is also straightforward.

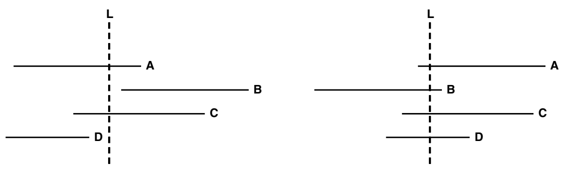


Figure 4: Illustrates the crucial role of row-convexity in path-consistent networks. The consecutive set of tick marks (consistent combinations) are represented as line segments. The left hand side shows that if there does not exist a point of overlap (indicated by a vertical line $L$) in a set of line segments, then the real reason for that is that some two of the line segments do not overlap ($B$ and $D$ do not intersect in the figure). The right hand side shows that if every two line segments overlap, then there is a common intersection point (indicated by line $L$). Put together, row-convexity implies global consistency in path-consistent networks.

| $C_1$ | $C_2$ | $C_3$ | $C_4$ | $C_5$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 |

| $C_1$ | $C_3$ | $C_2$ | $C_4$ | $C_5$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 |

Figure 5: Illustrates the fact that a (0,1)-matrix can be row-convex under one ordering of the columns of the matrix (left hand side), but not necessarily so under another ordering (right hand side).

1995) that if there exist domain orderings for the variables $X_1, X_2 \ldots X_N$ in a path-consistent network of binary constraints such that all the relations (constraints) can be made row-convex, then the network is globally consistent. (A *globally consistent* network has the property that a solution can be found in a backtrack-free manner.) Figure 3 roughly illustrates the underlying reasons for this claim. At any given point of time, suppose we have instantiated $k-1$ variables $X_1, X_2 \ldots X_{k-1}$, and suppose we are on the verge of instantiating the variable $X_k$. Having achieved path-consistency, and having chosen domain orderings for all the variables such that row-convexity is established, we notice that every instantiation $X_j = v_j$ $(1 \leq j \leq k-1)$ induces a continuous range of domain values (of $X_k$) that are consistent with $X_j = v_j$ $(1 \leq j \leq k-1)$. Because of this convexity, the lack of a consistent extension for variable $X_k$ implies that some two instantiations must have induced non-overlapping intervals (see Figure 4), and this conflict must have therefore also been detected while establishing path-consistency (variables $X_2$ and $X_4$ in Figure 3). Hence, if no inconsistency is detected while enforcing path-consistency (and if row-convexity holds), the network is guaranteed to be globally consistent.

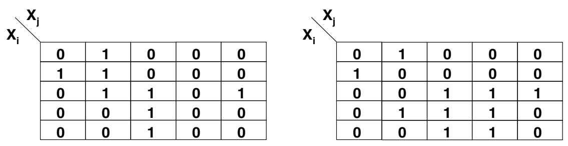
The orderings on the domain values of all the variables is critical to establishing row-convexity in path-consistent networks. For a single Boolean matrix (representing a con-

| | | $X_1$ | | | $X_2$ | | | $X_3$ | |
|---|---|---|---|---|---|---|---|---|---|
| | $d_{11}$ | $d_{12}$ | $d_{13}$ | $d_{21}$ | $d_{22}$ | $d_{23}$ | $d_{31}$ | $d_{32}$ | $d_{33}$ |
| $X_1$ | | | | | | | | | |
| $d_{11}$ | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 |
| $d_{12}$ | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| $d_{13}$ | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 |
| $X_2$ | | | | | | | | | |
| $d_{21}$ | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 |
| $d_{22}$ | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 |
| $d_{23}$ | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |
| $X_3$ | | | | | | | | | |
| $d_{31}$ | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| $d_{32}$ | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 |
| $d_{33}$ | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |

Figure 6: Illustrates how the algorithm in (Booth and Lueker 1976) can be applied to automatically find the domain orderings of all the variables so as to establish row-convexity in path-consistent networks (if such orderings exist). To find the domain ordering for variable $X_i$, we first stack up the matrix representations of all the constraints in which $X_i$ participates; we then apply the algorithm in (Booth and Lueker 1976) to find a permutation of the columns to achieve row-convexity in the stacked-up matrix (indicated by dotted boundaries for each variable).

straint), Figure 5 illustrates how row-convexity is apparent under one ordering (permutation) of the columns of the matrix, but not under another. However, in order to find the required domain orderings, we can make use of the following well-known result (Booth and Lueker 1976). Given an $m \times n$ Boolean matrix, we can find a permutation of the $n$ columns so that all the '1's appear consecutively in any row (if such a permutation exists) in $O(m+n+f)$ time. Here, $f$ is the total number of '1's in the matrix, and is $O(mn)$. The algorithm for doing this employs PQ-trees, and a detailed description of it can be found in (Booth and Lueker 1976). To find the domain orderings for all the variables so as to establish row-convexity in a path-consistent network (if it is possible to do so), we can therefore run the above algorithm once for each variable with the domain values of that variable representing the columns, and the different values of all other variables representing the rows (see Figure 6 and (Van Beek and Dechter 1995)).

Although row-convexity implies global consistency in path-consistent networks, the very process of achieving path-consistency may destroy it. (This means that row-convexity of the original set of constraints does not necessarily imply global consistency.) In particular, two problems arise while enforcing path-consistency on row-convex constraints. First, when a row-convex constraint is composed of disjoint blocks of '1's, its composition with another row-convex constraint may not be row-convex. Second, even if disjoint blocks are forbidden, intersection may create *empty* rows and columns (rows or columns that have only '0's in them) that lead to disjoint blocks. The following examples illustrate these problems (see (Deville *et al.* 1999)).



| | | | | |
|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 |

| | | | | |
|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 0 |

Figure 7: Illustrates the difference between row-convex and CRC constraints. The constraint on the left hand side is CRC because after deleting all the empty rows and columns (fourth column), the '1's appear contiguously in every row and every column; and moreover, the bands of '1's in consecutive rows touch (or overlap with) each other. The constraint on the right hand side is not CRC (although it is row-convex), because the bands of '1's in the second and third rows do not touch each other.

$$\begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix} \cap \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

CRC constraints avoid both the above problems by imposing a few additional restrictions. A (0,1)-matrix is CRC if, after removing empty rows and columns, it is row-convex and connected (i.e. the positions of the '1's in any two consecutive rows intersect, or are consecutive). A binary relation $R_{ij}$ constitutes a CRC constraint if both $M_{R_{ij}}$ and $M^T_{R_{ij}}$ are CRC. Unlike row-convex constraints, CRC constraints are closed under composition, intersection and transposition—the three basic operations employed by algorithms that enforce path-consistency—hence establishing path-consistency over CRC constraints is sufficient to ensure global consistency (see (Deville *et al.* 1999)). An instantiation of the generic path-consistency algorithm, that further exploits the structure of CRC constraints, has a running time complexity of $O(N^3K^2)$ and a space complexity of $O(N^2K)$ (see (Deville *et al.* 1999)). Here, $N$ is the number of variables, and $K$ is the size of the largest domain. (After path-consistency is achieved, a solution can be found in a backtrack-free manner in $O(N^2)$ time.) Figure 7 provides examples of row-convex and CRC constraints.

## Filtering with CRC Constraints

In a logical filtering scenario (see Figure 8), we essentially have to address the following combinatorial problem (a more formal definition of the logical filtering problem appears in (Amir and Russell 2003)). We are given a set of system variables $X_1^{(t)}, X_2^{(t)} \dots X_n^{(t)}$ (indexed by time $t$ because we are dealing with dynamical systems), where a complete assignment to the variables represents a state of the system at time $t$. The initial state of the system (at time 0) is specified in one of several forms, and the system evolves from time $t$ to time $t + 1$ under a specified transition model. The transition model is specified as a set of constraints between the variables $X_1^{(t)}, X_2^{(t)} \dots X_n^{(t)}$ and the variables $X_1^{(t+1)}, X_2^{(t+1)} \dots X_n^{(t+1)}$, and the observations are recorded on a subset (potentially empty) of the variables at every time step. The belief state at time $t$ is the set of all assignments to the variables $X_1^{(t)}, X_2^{(t)} \dots X_n^{(t)}$ that can be extended to a complete assignment (for all the variables across all time points $\leq t$) that is consistent with the initial state, observations, and the transition constraints (up to time $\leq t$). The goal is to maintain a compact representation of the belief state (at every time point) under an unbounded sequence of transitions and observations.

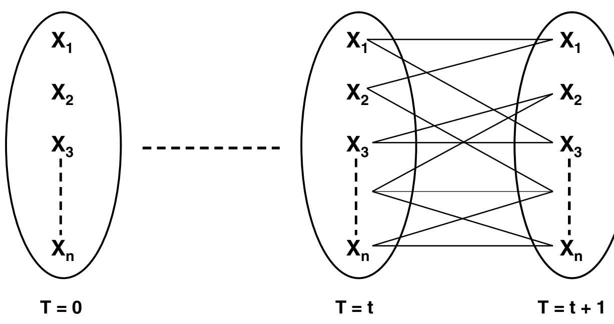


Figure 8: Illustrates the combinatorial problem in logical filtering. $X_1, X_2 \dots X_n$ (indexed by time $t$) are the system variables—values of which define a state of the system (at time $t$). Possible transitions of the system from time $t$ to time $t+1$ are defined using constraints between the variables at time $t$ and time $t + 1$. Observations on a subset (potentially empty) of the variables are recorded at every point of time; and the goal is to maintain (at every stage) a compact representation of the belief state.

In this section, we will show that efficient logical filtering is possible when all the constraints are CRC. From the previous section, we note that given a set of CRC constraints, enforcing path-consistency ensures global consistency. In other words, after path-consistency is enforced on a set of CRC constraints, any consistent instantiation of any subset of the variables (that satisfies all the direct constraints) can be extended to a complete solution (that assigns a value to all the variables and satisfies all the constraints). In turn, this means that given a set of CRC constraints over the variables $Y_1, Y_2 \dots Y_N$, the set of assignments to a subset of the variables $Y_{i_1}, Y_{i_2} \dots Y_{i_k}$ that can be extended to a complete solution are exactly those that satisfy the direct constraints between $Y_{i_1}, Y_{i_2} \dots Y_{i_k}$ after path-consistency is achieved.

In the context of logical filtering, if all the constraints are CRC, then the belief state at time $t$ is exactly the set of all solutions to the direct constraints between $X_1^{(t)}, X_2^{(t)} \dots X_n^{(t)}$ (after path-consistency is enforced). Therefore, one simple algorithm for logical filtering (over CRC constraints) is to repeatedly enforce path-consistency between the variables

```
ALGORITHM: CRC-FILTERING
INPUT: variables $X_1, X_2 \dots X_n$ (indexed by time $t$) with respective domains $D_1, D_2 \dots D_n$; CRC constraints $\mathcal{C}_I$ between the variables $X_1^{(t-1)}, X_2^{(t-1)} \dots X_n^{(t-1)}$ (solutions to which represent the belief state at time $t-1$); CRC constraints $\mathcal{C}_T$ between the variables $X_1^{(t-1)}, X_2^{(t-1)} \dots X_n^{(t-1)}$ and the variables $X_1^{(t)}, X_2^{(t)} \dots X_n^{(t)}$ representing possible transitions; observations $\mathcal{C}_O$ expressed as CRC constraints.
OUTPUT: constraints $\mathcal{C}$ between the variables $X_1^{(t)}, X_2^{(t)} \dots X_n^{(t)}$ (solutions to which represent the belief state at time $t$).
    (1) Establish path-consistency between all variables
        $\{X_1^{(t-1)}, X_2^{(t-1)} \dots X_n^{(t-1)}\} \cup \{X_1^{(t)}, X_2^{(t)} \dots X_n^{(t)}\}$
        using the constraints $\mathcal{C}_I \cup \mathcal{C}_T \cup \mathcal{C}_O$.
    (2) RETURN: the set of direct constraints between the
        variables $X_1^{(t)}, X_2^{(t)} \dots X_n^{(t)}$.
END ALGORITHM
```

Figure 9: Shows the algorithm for filtering CRC constraints. We assume that path-consistency over CRC constraints is enforced efficiently using the algorithm in (Deville *et al.* 1999). We also note that the set of direct constraints referred to in step (2) is the set of all binary (CRC) constraints between variables in $X_1^{(t)}, X_2^{(t)} \dots X_n^{(t)}$ after path-consistency is achieved.

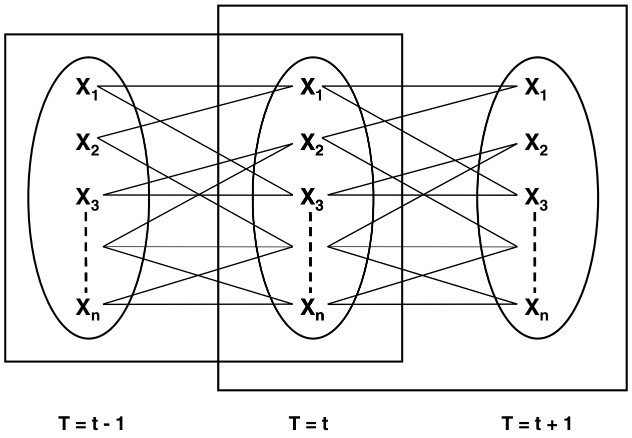
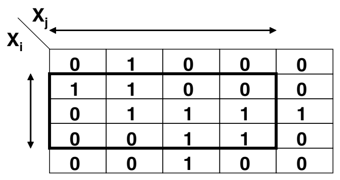


Figure 10: Presents a diagram to support some of the claims made in this section (when we are dealing with the problem of filtering CRC constraints). Establishing path-consistency between the variables at time $t-1$ and time $t$, and then between the variables at time $t$ and time $t+1$, ensures—by the Markovian property—that a consistent assignment to the variables at time $t+1$ has a consistent extension to all the previous variables.



| | | | | |
|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 |

Figure 11: Shows that a CRC constraint restricted to any continuous range of values of any participating variable induces a CRC constraint in turn. The line segments indicate the restriction of the domains of the variables $X_i$ and $X_j$ to continuous ranges of values, and the rectangle with dark edges indicates the induced CRC constraint.

in successive time steps, and record only the direct constraints between the variables at the current time step. Figure 9 presents this algorithm for logical filtering when all the relevant constraints are CRC.

We note that every time path-consistency is achieved between the variables $X_1^{(t-1)}, X_2^{(t-1)} \dots X_n^{(t-1)}$ and $X_1^{(t)}, X_2^{(t)} \dots X_n^{(t)}$ using the constraints $\mathcal{C}_I \cup \mathcal{C}_T \cup \mathcal{C}_O$ (see Figure 9), only the resulting direct constraints between the variables $X_1^{(t)}, X_2^{(t)} \dots X_n^{(t)}$ are retained to represent the belief state at time $t$. In the next time step, the same process is repeated for the variables $X_1^{(t)}, X_2^{(t)} \dots X_n^{(t)}$ and $X_1^{(t+1)}, X_2^{(t+1)} \dots X_n^{(t+1)}$. At this stage (time $t+1$), it looks as if path-consistency must be achieved over all the variables across all time steps $\leq t+1$ (including $X_1^{(t-1)}, X_2^{(t-1)} \dots X_n^{(t-1)}$) for justifying the retention of only the CRC constraints over $X_1^{(t+1)}, X_2^{(t+1)} \dots X_n^{(t+1)}$ to represent the belief state at time $t+1$. However, the Markovian nature of the evolution of the system helps us in proving the soundness of the procedure in Figure 9 (that considers establishing path-consistency only over the variables in $X_1^{(t)}, X_2^{(t)} \dots X_n^{(t)}$ and $X_1^{(t+1)}, X_2^{(t+1)} \dots X_n^{(t+1)}$). In particular, establishing path-consistency between the variables at time $t-1$ and time $t$ ensures that any consistent assignment of values to the variables $X_1^{(t)}, X_2^{(t)} \dots X_n^{(t)}$ can also be consistently extended to an assignment for all the previous variables; and moreover, when path-consistency is achieved between the variables $X_1^{(t)}, X_2^{(t)} \dots X_n^{(t)}$ and $X_1^{(t+1)}, X_2^{(t+1)} \dots X_n^{(t+1)}$, it can only constrain the set of consistent assignments to the variables $X_1^{(t)}, X_2^{(t)} \dots X_n^{(t)}$; and any consistent assignment to the variables $X_1^{(t+1)}, X_2^{(t+1)} \dots X_n^{(t+1)}$ can be consistently extended to $X_1^{(t)}, X_2^{(t)} \dots X_n^{(t)}$. Put together, any consistent assignment to the variables $X_1^{(t+1)}, X_2^{(t+1)} \dots X_n^{(t+1)}$ (that satisfies the direct constraints) can also be consistently extended to all the previous variables (see Figure 10).

We will now look at ways in which the initial state, transition model and the observations can be specified so that all the resulting constraints turn out to be CRC.

**Initial State:** Uncertainty in the initial state can be handled when the possible initial states are expressed as the set of all solutions to a set of CRC constraints over $X_1^{(0)}, X_2^{(0)} \dots X_n^{(0)}$. As a trivial consequence, if there is no uncertainty in the initial state, this definite state (say $X_1^{(0)} = v_1, X_2^{(0)} = v_2 \dots X_n^{(0)} = v_n$) can be expressed as a set of CRC constraints where every binary CRC constraint between $X_i^{(0)}$ and $X_j^{(0)}$ has only one allowed combination: $X_i^{(0)} = v_i$ and $X_j^{(0)} = v_j$. Further, if uncertainty is in the form of $k$ different initial states that cannot be expressed as solutions to a set of CRC constraints (and if all the other constraints are CRC), then the resulting filtering problem is equivalent to $k$ different filtering problems each with a different (but fixed) initial state.

**Transition Model:** The transition model speaks about how a system evolves over time (perhaps under the influence of actions taken by the agent). A set of transition constraints between the variables $X_1^{(t)}, X_2^{(t)} \dots X_n^{(t)}$ and the variables $X_1^{(t+1)}, X_2^{(t+1)} \dots X_n^{(t+1)}$ specifies whether a transition is possible from state $s$ at time $t$ (expressed as an assignment to the variables $X_1^{(t)}, X_2^{(t)} \dots X_n^{(t)}$) to state $s'$ at time $t+1$ (expressed as an assignment to the variables $X_1^{(t+1)}, X_2^{(t+1)} \dots X_n^{(t+1)}$). CRC constraints can be used to express the necessary transitions in some useful domains (see later section for an example).

**Observations:** In the simplest case where observations are expressed as values for certain variables, the resulting constraints are certainly CRC (see Figure 11). However, we can allow for a much richer representation of observations; in particular, if uncertainty in the observations is expressed as continuous ranges of possible domain values to certain variables, then the induced constraints would still remain CRC (see Figure 11).

We note that at any time $t$, the set of possible states of the system is represented as the set of all solutions to the direct CRC constraints over the variables $X_1^{(t)}, X_2^{(t)} \dots X_n^{(t)}$. Since there are at most $O(n^2)$ such constraints, the representation is compact (and indefinitely so because it is independent of $t$). We also note that the time required to update the belief state is $O(n^3K^2)$ and the space required to specify the belief state is $O(n^2K^2)$ (where $K$ is the size of the largest domain). These complexities are directly analogous to that of Kalman filtering for linear Gaussian models.

Our approach differs from approaches that maintain the belief state at time $t$ as a set of complete assignments to the system variables $X_1^{(t)}, X_2^{(t)} \dots X_n^{(t)}$. In particular, approaches of the latter kind quickly run into the worst case scenario of having to represent an exponential number of states achievable at time $t$. Our approach, however, represents these exponential number of states achievable at time $t$ as the set of all solutions to a set of only $O(n^2)$ CRC constraints (if all the constraints in the system are CRC). Further, most reasoning tasks on this representation are easy; in particular, it is easy to verify whether a state (specified as an assignment to the variables $X_1^{(t)}, X_2^{(t)} \dots X_n^{(t)}$) is achievable at time $t$ (i.e., satisfies all the direct CRC constraints between the variables $X_1^{(t)}, X_2^{(t)} \dots X_n^{(t)}$), and it is also easy to obtain a candidate state at time $t$ by efficiently solving the CRC constraints between the variables $X_1^{(t)}, X_2^{(t)} \dots X_n^{(t)}$ (simple randomized algorithms for solving CRC constraints can be found in (Kumar 2005b)). Finally, our approach also differs from approaches that try to maintain the belief state at time $t$ using ellipsoids or bounding hyperplanes (El Ghaoui and Calafiore 1999). In particular, there are very simple cases where the solutions to a set of CRC constraints cannot be represented compactly using geometrically closed regions (e.g. RDTPs (Kumar 2005a)).

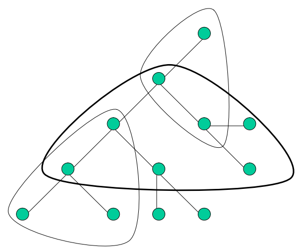


Figure 12: Illustrates the fact that in a tree structure, if a set of subtrees do not have a common node, then there exist some two subtrees that do not overlap with each other (the subtrees within lighter enclosures in the figure). Conversely, if every two subtrees have at least one node in common, then there exists a node that is common to all the subtrees. (The enclosed areas in the figure indicate subtrees.)

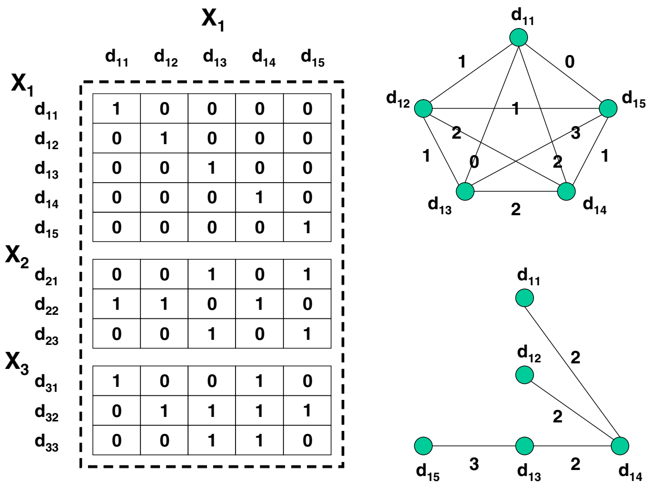
```
ALGORITHM: TREE-EMBEDDING
INPUT: variables $Z_1, Z_2 \dots Z_N$ with respective domains $F_1, F_2 \dots F_N$; binary constraints $C_1, C_2 \dots C_M$ (the binary constraint between $Z_i$ and $Z_j$ is denoted by $C_{ij}$).
OUTPUT: tree structures $T_i$ ($i = 1, 2 \dots N$) on the domain values of each variable.
  (1) For every variable $Z_i$:
    (a) Construct complete graph $G$ on the domain values $f_1, f_2 \dots f_{|F_i|} \in F_i$ of $Z_i$:
    (b) For the undirected edge $(f_a, f_b)$ assign a weight $= |\{(Z_j, v) : v \in F_j \cap C_{ij}(f_a, v) \cap C_{ij}(f_b, v)\}|$.
    (c) Construct $T_i$ = the maximum spanning tree of $G$.
  (2) RETURN: $T_1, T_2 \dots T_N$.
END ALGORITHM
```

Figure 13: A simple polynomial time algorithm for constructing a tree $T_i$ (if such a tree exists) over a set of domain values $F_i$ of the variable $Z_i$ such that for any value of any other variable, the set of values in $F_i$ that are consistent with it constitutes a single subtree in $T_i$.

## Filtering with Path-Consistency

In this section, we will extend the theory developed in the previous section (for filtering CRC constraints) to perform filtering in the more general context of tree-convexity (Zhang and Freuder 2004). In particular, we will provide a filtering algorithm based on repeatedly enforcing path-consistency and embedding the domain values of the related variables in tree structures (with certain properties) to be able to guarantee global consistency.

From the previous section, we know that path-consistency implies global consistency when there exist orderings on the domain values of all the variables that establish row-



| | $d_{11}$ | $d_{12}$ | $d_{13}$ | $d_{14}$ | $d_{15}$ |
|---|---|---|---|---|---|
| $X_1$ $d_{11}$ | 1 | 0 | 0 | 0 | 0 |
| $d_{12}$ | 0 | 1 | 0 | 0 | 0 |
| $d_{13}$ | 0 | 0 | 1 | 0 | 0 |
| $d_{14}$ | 0 | 0 | 0 | 1 | 0 |
| $d_{15}$ | 0 | 0 | 0 | 0 | 1 |
| $X_2$ $d_{21}$ | 0 | 0 | 1 | 0 | 1 |
| $d_{22}$ | 1 | 1 | 0 | 1 | 0 |
| $d_{23}$ | 0 | 0 | 1 | 0 | 1 |
| $X_3$ $d_{31}$ | 1 | 0 | 0 | 1 | 0 |
| $d_{32}$ | 0 | 1 | 1 | 1 | 1 |
| $d_{33}$ | 0 | 0 | 1 | 1 | 0 |

Figure 14: Illustrates the working of the algorithm in Figure 13. The left side of the figure shows the binary constraints between $X_1$ and every other variable in the form of Boolean matrices. The right side of the figure shows the maximum spanning tree for the complete graph constructed over the domain values of $X_1$.

convexity of all the constraints. A key observation that was exploited in justifying this claim is that given a set of line segments, if every two line segments overlap, then there is a common intersection point (see Figure 4). An extension of this observation is that trees exhibit a similar behavior (see Figure 12). That is, if there does not exist a node that is common to all the subtrees of a tree (from a given collection of subtrees), then there exist some two subtrees that do not overlap. Conversely, if every two subtrees have at least one node in common, then there exists a node that is common to all the subtrees. In the context of filtering, this means that in going from time $t-1$ to time $t$ (after path-consistency is achieved), if we can embed the domain values $D$ of every variable $X \in \{X_1^{(t-1)}, X_2^{(t-1)} \ldots X_n^{(t-1)}\} \cup \{X_1^{(t)}, X_2^{(t)} \ldots X_n^{(t)}\}$ in a tree structure $T$ such that for any value of any other variable, the set of values in $D$ that are consistent with it constitutes a single subtree in $T$, then such an embedding provides a certificate for global consistency.[2]

Figure 15 provides an algorithm (generalizing that of Figure 9) for filtering binary constraints by repeatedly enforcing path-consistency (at appropriate steps) and testing its sufficiency for ensuring global consistency. (As before, if global consistency is true, then the belief state at time $t$ is given by the set of direct binary constraints between the variables in $X_1^{(t)}, X_2^{(t)} \ldots X_n^{(t)}$ after path-consistency is achieved.) Figure 13 provides a simple polynomial time algorithm for automatically constructing the required tree embeddings (if they exist) of the domain values of all the variables to verify whether a path-consistent network can be made tree-convex

```
**ALGORITHM:** FILTERING-WITH-PC
**INPUT:** variables $X_1, X_2 \ldots X_n$ (indexed by time $t$) with respective domains $D_1, D_2 \ldots D_n$; constraints $\mathcal{C}_I$ between the variables $X_1^{(t-1)}, X_2^{(t-1)} \ldots X_n^{(t-1)}$ (solutions to which represent the belief state at time $t-1$); binary constraints $\mathcal{C}_T$ between the variables $X_1^{(t-1)}, X_2^{(t-1)} \ldots X_n^{(t-1)}$ and the variables $X_1^{(t)}, X_2^{(t)} \ldots X_n^{(t)}$ representing possible transitions; observations expressed as constraints $\mathcal{C}_O$.
**OUTPUT:** constraints $\mathcal{C}$ between the variables $X_1^{(t)}, X_2^{(t)} \ldots X_n^{(t)}$ (solutions to which represent the belief state at time $t$).
  (1) Establish path-consistency between all variables $\{X_1^{(t-1)}, X_2^{(t-1)} \ldots X_n^{(t-1)}\} \cup \{X_1^{(t)}, X_2^{(t)} \ldots X_n^{(t)}\}$ using the constraints $\mathcal{C}_I \cup \mathcal{C}_T \cup \mathcal{C}_O$.
  (2) Embed the domain values of all the variables $X_1^{(t-1)}, X_2^{(t-1)} \ldots X_n^{(t-1)}$ and $X_1^{(t)}, X_2^{(t)} \ldots X_n^{(t)}$ in tree structures using the algorithm in Figure 13.
  (3) If for any $X \in \{X_1^{(t-1)}, X_2^{(t-1)} \ldots X_n^{(t-1)}\} \cup \{X_1^{(t)}, X_2^{(t)} \ldots X_n^{(t)}\}$, and any value of any other variable, the set of domain values of $X$ consistent with it constitutes a single subtree:
    (a) RETURN: the set of direct constraints between the variables $X_1^{(t)}, X_2^{(t)} \ldots X_n^{(t)}$.
  (4) Else RETURN: failure.
**END** ALGORITHM
```

Figure 15: Shows the algorithm for filtering binary constraints using path-consistency. The success of the algorithm depends crucially on whether we can embed the domain values of the variables in appropriate tree structures.

in order to provide a certificate of global consistency. Our algorithm is a very simple adaptation of an algorithm presented in (Conitzer *et al.* 2004) in the context of solving combinatorial auctions; and is (in some sense) a generalization of the algorithm in (Booth and Lueker 1976) for identifying row-convexity (i.e., the *consecutive ones* property).

Given a path-consistent network, consider the task of constructing a tree $T_i$ (over the domain values in $F_i$) for the variable $Z_i$ such that for any value $v$ of any other variable $Z_j$, the set of values in $F_i$ that are consistent with it constitutes a single subtree in $T_i$. For every value $v$ of every other variable $Z_j$, we construct the set $S_{Z_j,v} \subseteq F_i$ of values in $F_i$ that are consistent with it. Now, given a set of such subsets $S_1, S_2 \ldots S_k$ (one for each value of every other variable), we construct a weighted undirected graph as follows. The nodes of the graph correspond to the values in $F_i$, and an undirected edge between two nodes is assigned a weight equal to the number of subsets $S_1, S_2 \ldots S_k$ in which the corresponding two values occur together. The required tree $T_i$ (if it exists) is then given by the *maximum weighted spanning tree* on this graph (which can be computed very efficiently). A rigorous proof for this claim is presented in (Conitzer *et al.* 2004). Figure 14 illustrates the working of this algorithm (see Figure 13).

[2]Note that the tree structure for a variable at time $t$ can be different from that at time $t+1$ because the tree structures are used only to provide certificates of global consistency.

```
ALGORITHM: FILTERING-WITH-LC
INPUT: variables $X_1, X_2 \dots X_n$ (indexed by time $t$) with respective domains $D_1, D_2 \dots D_n$; constraints $\mathcal{C}_I$ between the variables $X_1^{(t-1)}, X_2^{(t-1)} \dots X_n^{(t-1)}$ (solutions to which represent the belief state at time $t-1$); constraints $\mathcal{C}_T$ between the variables $X_1^{(t-1)}, X_2^{(t-1)} \dots X_n^{(t-1)}$ and the variables $X_1^{(t)}, X_2^{(t)} \dots X_n^{(t)}$ representing possible transitions; observations expressed as constraints $\mathcal{C}_O$.
OUTPUT: constraints $\mathcal{C}$ between the variables $X_1^{(t)}, X_2^{(t)} \dots X_n^{(t)}$ (solutions to which represent the belief state at time $t$).
  (1) For $k = 3 \dots n$:
    (a) Establish $k$-consistency between all variables $X_1^{(t-1)}, X_2^{(t-1)} \dots X_n^{(t-1)}$ and $X_1^{(t)}, X_2^{(t)} \dots X_n^{(t)}$ using the constraints $\mathcal{C}_I \cup \mathcal{C}_T \cup \mathcal{C}_O$.
    (b) Construct geometric embeddings of the domain values of all the variables $X_1^{(t-1)}, X_2^{(t-1)} \dots X_n^{(t-1)}$ and $X_1^{(t)}, X_2^{(t)} \dots X_n^{(t)}$ to provide a certificate of global consistency.
    (c) If such embeddings exist:
      (i) RETURN: the set of direct constraints between the variables $X_1^{(t)}, X_2^{(t)} \dots X_n^{(t)}$.
    (d) Else $k = k + 1$.
END ALGORITHM
```

Figure 16: Shows the algorithm for filtering a set of constraints by enforcing increasing levels of local consistency. At each stage (level of local consistency), we check whether geometric embeddings of the domain values of all the variables exist so as to provide a certificate for global consistency.

## Filtering with Local Consistency

In this section, we will comment on the possible extension of the theory developed in the previous section to perform filtering in the general case. Figure 16 presents the algorithm for doing this. The key idea is the observation that, in general, we need to enforce $n$-consistency between the variables $X_1^{(t-1)}, X_2^{(t-1)} \dots X_n^{(t-1)}$ and $X_1^{(t)}, X_2^{(t)} \dots X_n^{(t)}$ to update the belief state from stage $t-1$ to stage $t$; however, if increasing levels of local consistency are achieved progressively, and $k$-consistency (for some $k$ significantly less than $n$) can be shown to imply global consistency, then no further work needs to be done—hence saving the effort of having to achieve $n$-consistency. (We note that establishing $k$-consistency ($n$-consistency) requires computational resources proportional to $K^k$ ($K^n$) where $K$ is the size of the largest domain.)

A fundamental combinatorial problem arising in the above scheme is related to identifying the conditions under which we can prove that $k$-consistency implies global consistency. The ideas of row- and tree-convexity, for example, were used for $k = 3$. For higher $k$, two important questions need to be answered: "What kind of a geometric embedding of the domain values of all the variables ensures global consistency?"and "How do we efficiently find such embeddings when they exist?" (i.e. "What is the algorithm analogous to that in Figure 13?"). We surmise that, in general, the problem is related to embedding the domain values of variables in *clique trees*—although how to construct them automatically is a largely open question.[3]

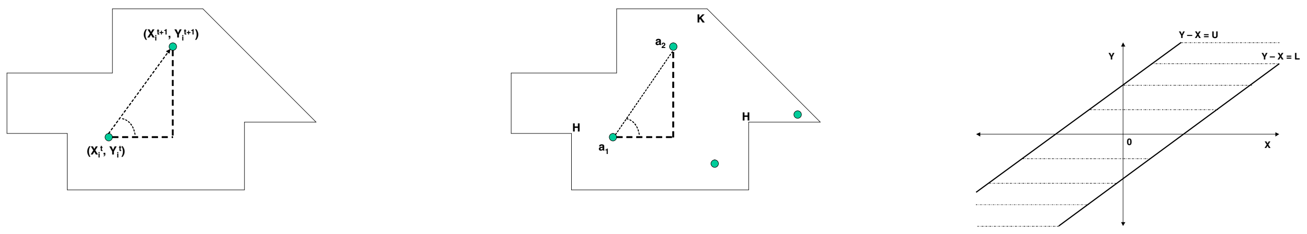
## Example: Multi-Agent Localization in a CRC Region

In this section, we will present an application of the theory developed in this paper. The scenario involves multiple agents (robots) that have to collaborate in achieving a certain task, and in the process have to localize in a certain region by communicating with each other and making limited observations of their current states. Figure 17 and Figure 18 help to illustrate this scenario. There are $N$ agents $a_1, a_2 \dots a_N$, with the position of agent $a_i$ described by a pair of coordinates $(x_i^{(t)}, y_i^{(t)})$ at any time $t$. (For simplicity, we assume a grid world instead of allowing $x_i^t$ and $y_i^t$ to be continuous.)

An agent $a_i$ can make a strategic move at time $t$ to achieve a new position at time $t+1$. The agent keeps track of its displacements; and in particular, it approximately measures the distance it has moved, and the angle along which it did so at any time $t$. From this, the agent might infer that $L_1^{(t)} \leq x_i^{(t+1)} - x_i^{(t)} \leq U_1^{(t)}$ and $L_2^{(t)} \leq y_i^{(t+1)} - y_i^t \leq U_2^{(t)}$ (for some constants $L_1^{(t)}$, $U_1^{(t)}$, $L_2^{(t)}$ and $U_2^{(t)}$). Agents $a_i$ and $a_j$ can also communicate approximate mutual distances and orientations with each other to infer $L_3^{(t)} \leq x_i^{(t)} - x_j^{(t)} \leq U_3^{(t)}$ and $L_4^{(t)} \leq y_i^{(t)} - y_j^{(t)} \leq U_4^{(t)}$ (for some constants $L_3^{(t)}$, $U_3^{(t)}$, $L_4^{(t)}$ and $U_4^{(t)}$). As shown in Figure 17(c), all these kinds of constraints are CRC. Moreover, the constraint that the coordinates of any agent must be within the boundaries of the region in which it is trying to localize is also a CRC constraint if we assume that the geometry of the region is CRC. (Many realistic domains, such as football fields, qualify as CRC regions.)

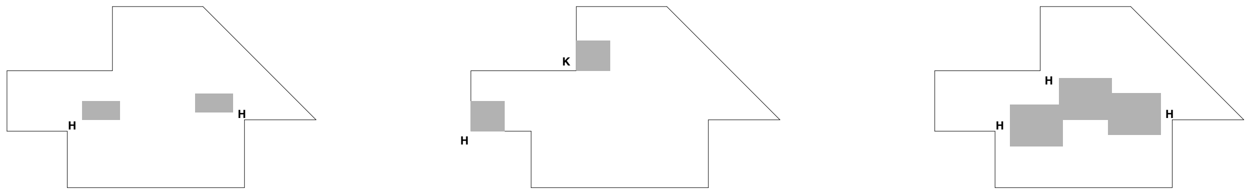
Finally, the presence of landmarks helps agents in localization; but may or may not create CRC constraints that are useful for filtering. Figure 18 discusses some of these cases. If there is a landmark $H$, and if an agent is near it (enabling it to localize in a small rectangular region around it), then the resulting information constitutes a CRC region (constraint); moreover, when there are two such landmarks that look alike, the resulting disjunctive information also constitutes a CRC constraint (see Figure 18(a)). Sometimes, even the absence of a nearby landmark can help an agent in localization (removing the shaded area near the landmark $H$ results in a CRC region in Figure 18(b)), but not always (removing the shaded area near the landmark $K$ does not result in a CRC region in Figure 18(b)). Finally, when there are multiple landmarks of the same kind, the resulting constraining region may not always be CRC (the union of the shaded areas representing regions of localization with respect to each landmark is not CRC in Figure 18(c)).

[3]This is different from constructing clique trees over the *variables* as in standard dynamic programming-based approaches.



(a) (b) (c)

Figure 17: Diagrams to illustrate the scenario of multi-agent localization in a CRC region. (a) shows the possible style in which an agent $a_i$ keeps track of its displacements; at any stage, it approximately measures the distance it has moved, and the angle along which it moved. (b) shows a CRC region and the possible style of communication between two agents $a_1$ and $a_2$; the two agents can communicate approximate distances and relative orientations with each other. (c) shows that any constraint of the form $L \leq Y - X \leq U$ is a CRC constraint (dotted lines indicate the feasible region).



(a) (b) (c)

Figure 18: Illustrates the role of landmarks in the scenario of Figure 17. (a) shows that if there is a landmark $H$, and if an agent is near it (enabling it to localize in a small rectangular region around it), then the resulting information constitutes a CRC region (constraint); moreover, when there are two such landmarks that look alike, the resulting disjunctive information also constitutes a CRC constraint. (b) shows that, sometimes, even the absence of a nearby landmark can help an agent in localization (removing the shaded area near the landmark $H$ results in a CRC region), but not always (removing the shaded area near the landmark $K$ does not result in a CRC region). (c) shows that when there are multiple landmarks of the same kind, the resulting constraining region may not always be CRC (the union of the shaded areas representing regions of localization with respect to each landmark is not CRC).

## Related Work, Conclusions, Future Work and Acknowledgements

Early work on filtering in the logical context includes the following: (Fikes *et al.* 1972), (Lin and Reiter 1997) (cases when actions have deterministic effects), (Liberatore 1997) (cases when the initial state is not fully known, or when actions have nondeterministic effects). Traditionally, computational approaches for filtering have taken one of three approaches: (1) enumerate the world states possible in every belief state and update each of those states separately, together generating the updated belief state (see (Ferraris and Giunchiglia 2000) and (Cimatti and Roveri 2000)), (2) list the sequence of actions and observations, and prove queries on the updated belief state (see (Reiter 2001) and (Sandewall 1994)), or (3) approximate the belief state representation (Son and Baral 2001). The first two approaches cannot be used when there are too many possible worlds, or when the sequence of actions is long. The third approach too poses the problem of giving rise to potentially unsafe situations. Filtering algorithms for actions that permute the state space, or when the belief state is represented using prime implicates (and under some assumptions), are presented in (Amir and Russell 2003). First-order logical filtering is also analyzed in (Shirazi and Amir 2005), and the projection problem (in the presence of context-dependent actions and incomplete first-order knowledge) is studied in (Liu and Levesque 2005).

In this paper, we identified classes of transition constraints that are amenable to compact and indefinite filtering—presenting efficient algorithms wherever necessary. We first showed that CRC constraints are amenable to efficient filtering when path-consistency is enforced in appropriate steps. We then extended this theory to provide a filtering algorithm based on repeatedly enforcing path-consistency and embedding the domain values of the related variables in tree structures to guarantee global consistency. (In turn, we alluded to possibly generalizing this theory to perform filtering via repeated enforcement of levels of local consistency followed by appropriate geometric embeddings of the domain values of the related variables to guarantee global consistency.) Finally, we identified and commented on the problem of multi-agent localization as a potential application of the theory developed in the paper (under some reasonable assumptions).

As part of our future work, we are interested in a more elaborate theory for characterizing tractable cases of logical filtering—whether or not they are related to geometric embeddings. We are also interested in dealing with CRC constraints in continuous domains (as in the case of multi-agent localization) using graphical representations such as *distance graphs* (Kumar 2005a).

Some of the above references to related work and some of the introductory material in this paper are borrowed from (Amir and Russell 2003).

## References

Amir E. and Russell S. 2003. Logical Filtering. *Proceedings of IJCAI'2003.*

Booth K. S. and Lueker G. S. 1976. Testing for the Consecutive Ones Property, Interval Graphs, and Graph Planarity Using PQ-tree Algorithms. *Journal of Computer and System Sciences, 13:335–379, 1976.*

Bryce D. and Kambhampati S. 2005. Cost Sensitive Reachability Heuristics for Handling State Uncertainty. *Proceedings of UAI'2005.*

Cimatti A. and Roveri M. 2000. Conformant Planning via Symbolic Model Checking. *JAIR, 13:305–338, 2000.*

Conitzer V., Derryberry J. and Sandholm T. 2004. Combinatorial Auctions with Structured Item Graphs. *Proceedings of AAAI'2004.*

Cushing W. and Bryce D. 2005. State Agnostic Planning Graphs (and Their Application to Belief Space Planning). *Proceedings of AAAI'2005.*

Deville Y., Barette O. and Van Hentenryck P. 1999. Constraint Satisfaction over Connected Row-Convex Constraints. *Artificial Intelligence, 109:243–271.*

El Ghaoui L. and Calafiore G. 1999. Confidence Ellipsoids for Uncertain Linear Equations with Structure. *Proc. Conf. Decision and Control, December 1999.*

Ferraris P. and Giunchiglia E. 2000. Planning as Satisfiability in Nondeterministic Domains. *Proceedings of AAAI'2000.*

Fikes R., Hart P. and Nilsson N. 1972. Learning and Executing Generalized Robot Plans. *AIJ, 3:251–288, 1972.*

Kalman R. E. 1960. A New Approach to Linear Filtering and Prediction Problems. *Trans. of ASME J. of Basic Engineering, 82(Ser. D):35–45, 1960.*

Kumar T. K. S. 2005a. On the Tractability of Restricted Disjunctive Temporal Problems. *Proceedings of the Fifteenth International Conference on Automated Planning and Scheduling (ICAPS'2005).*

Kumar T. K. S. 2005b. On the Tractability of Smooth Constraint Satisfaction Problems. *Proceedings of the Seventh International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CP-AI-OR'2005).*

Liberatore P. 1997. The Complexity of the Language A. *ETAI, 1:13–38, 1997.*

Lin F. and Reiter R. 1997. How to Progress a Database. *AIJ, 92:131–167, 1997.*

Liu Y. and Levesque H. 2005. Tractable Reasoning with Incomplete First-Order Knowledge in Dynamic Systems with Context-Dependent Actions. *Proceedings of IJCAI'2005.*

Mohr R. and Henderson T. C. 1986. Arc and Path Consistency Revisited. *Artificial Intelligence, 28:225–233.*

Nguyen X. and Kambhampati S. 2000. Extracting Effective and Admissible Heuristics from the Planning Graph. *Proceedings of AAAI'2000.*

Reiter R. 2001. *Knowledge in Action.* MIT Press, 2001.

Sandewall E. 1994. *Features and Fluents.* Oxford, 1994.

Shirazi A. and Amir E. 2005. First-Order Logical Filtering. *Proceedings of IJCAI'2005.*

Son T. C. and Baral C. 2001. Formalizing Sensing Actions: A Transition Function Based Approach. *AIJ, 125.*

Van Beek P. and Dechter R. 1995. On the Minimality and Global Consistency of Row-Convex Constraint Networks. *Journal of the ACM, Volume 42, Issue 3, Pages: 543–561.*

Zhang Y. and Freuder E. C. 2004. Tractable Tree Convex Constraint Networks. *Proceedings of AAAI'2004.*