2014 IEEE International Conference on Robotics & Automation (ICRA)
Hong Kong Convention and Exhibition Center
May 31 - June 7, 2014. Hong Kong, China

More Knowledge on the Table: Planning with Space, Time and Resources for Robots

Masoumeh Mansouri[1] and Federico Pecora[1]

***Abstract*— AI-based solutions for robot planning have so far focused on very high-level abstractions of robot capabilities and of the environment in which they operate. However, to be useful in a robotic context, the model provided to an AI planner should afford both symbolic and metric constructs; its expressiveness should not hinder computational efficiency; and it should include causal, spatial, temporal and resource aspects of the domain. We propose a planner grounded on well-founded constraint-based calculi that adhere to these requirements. A proof of completeness is provided, and the flexibility and portability of the approach is validated through several experiments on real and simulated robot platforms.**

I. Introduction

A major contribution of AI to Robotics is the model-centered approach, whereby competent robot behavior stems from automated reasoning in models of the world which can be changed to suit different environments, physical capabilities, and tasks. When it comes to implementing robot behaviors, the models used must be convenient to specify and lead to executable plans. They must maximize portability, so as to reduce the amount of modeling necessary for deployment on different robot platforms and/or different environments. These requirements imply, on one hand, that models should be abstract and, to the extent possible, qualitative. For example, a qualitative spatial model stating that knives should be placed "to the right" of dishes and forks "to the left". On the other hand, symbolic relations should subsume metric knowledge that can be used for actuation, and the use of model-based reasoning should not exclude the possibility of providing explicit metric specifications to facilitate robot programming. Furthermore, the expressiveness of the model should not come at the cost of inefficient reasoning, and it should be possible to employ well-established calculi to enable the use of off-the-shelf reasoners.

In order to derive robot behavior from models, these should represent diverse aspects of the domain in which the robot operates: some are related to the causality between actions, others to temporal aspects of the domain, spatial characteristics of the scenario, and objects the robot manipulates. There exist formalisms for representing many of these aspects, and reasoning methods have been developed for each individually. However, a loose coupling of specialized reasoners is not alone sufficient for dealing with the complexities of the real world. For instance, setting a well-set table requires more than spatial reasoning: fork and knife may be too close for placing the dish between them (spatial reasoning), therefore requiring to plan actions for making space (causal reasoning); the plan would depend on how many arms the robot has, which in turn may require to schedule the use of each arm (resource and temporal reasoning). Interdependencies between spatial, causal, resource and temporal decisions must be taken into account in order to guarantee that a feasible course of action is found.

We propose a novel approach for planning with integrated spatial, temporal and resource models. The modeling language used to specify the planning domain combines well-founded constraint-based calculi that adhere to the requirements listed above. A novel spatial knowledge representation calculus is employed to accommodate hybrid qualitative and metric reasoning in 2D space, called ARA$^+$. The plans obtained are guaranteed to adhere to all modeled knowledge, and a proof of completeness of the planning algorithm is provided. We conclude the paper with several experiments on real and simulated robot platforms in a table setting domain.

II. Representation

Our approach is grounded on the notion of *fluent*. Fluents represent parts of real world that are relevant for robot decision making. These include the actuation and sensing capabilities of robotic systems, as well as other meaningful aspects of the environment. For instance, a fluent can be used to represent the robot actions "navigating" and "grasping". Similarly, a fluent can represent the interesting states of the environment, e.g., the state of a "cup" being "on" a "table". Let $\mathcal{F}$ be the set of fluents in a given application scenario.

Some fluents model physical devices which require resources. We employ the concept of *reusable resource*, i.e., a resource with a limited capacity which is fully available when not required by a device. For example, a reusable resource can be used to model the "grasping capacity" of a robot: if the robot has one arm, the capacity of this resource is one, modeling the fact that the robot can only hold one object at a time; capacity two would instead signify that the robot can hold two objects at a time, e.g, if it possesses two arms. We denote with $\mathcal{R}$ the set of all resource identifiers. Given a resource $R \in \mathcal{R}$, its capacity is a value $\mathrm{Cap}(R) \in \mathbb{N}$.

Some fluents assert spatial properties on one or more spatial entities. All spatial entities are modeled as bounded rectangles whose sides are parallel to the axes of the reference frame of their (common) support plane. A bounded rectangle is a set of four bounds $[l_x^1, u_x^1][l_y^1, u_y^1][l_x^2, u_x^2][l_y^2, u_y^2]$, where $[l_x^1, u_x^1][l_y^1, u_y^1]$ bound the position of the lower left corner (x^1, y^1), while $[l_x^2, u_x^2][l_y^2, u_y^2]$ bound the position of the upper right corner (x^2, y^2).

[1] Center for Applied Autononous Sensor Systems, Örebro University, 70182 Sweden — {mmi,fpa}@aass.oru.se

Definition 1: A *fluent* f is a tuple $(\mathrm{P}, \theta, I, u, br)$, where

- P is a first-order formula represented as $\mathrm{P}(\mathrm{t}_1, \ldots, \mathrm{t}_n)$ where P is a *predicate symbol* and $t_1, \ldots, t_n$ are *terms*;
- $\theta = \{t_1/v_1, \ldots, t_n/v_n\}$ is a *substitution* for the variable terms of the formula P; a fluent is *ground* if $\theta \neq \emptyset$ and $|\theta| = n$;
- $I = [I_s, I_e]$ is a *flexible temporal interval* within which P evaluates to `true`, where $I_s = [l_s, u_s], I_e = [l_e, u_e], l_{s/e}, u_{s/e} \in \mathbb{N}$ represent, respectively, an interval of admissibility of the start and end times of the fluent;
- $u : \mathcal{R} \rightarrow \mathbb{N}$ specifies the *resources used* by the fluent ($\emptyset$ if the fluent does not prescribe resource usage);
- br represents a bounded rectangle of a spatial entity ($\emptyset$ if the fluent does not assert spatial semantics).

In the following, we omit writing the substitution θ explicitly, and employ the notations $?\mathrm{t}$ and t to refer to non-ground and ground predicates, respectively. For example, the fluent

$$f_1 = (\mathrm{Pickup}(\mathrm{cup1}, \mathrm{table1}), [[10, 10], [30, 50]], u(\mathrm{arm}) = 1, \emptyset)$$

represents the temporal fact that the robot picks up $\mathrm{cup1}$ from $\mathrm{table1}$ in a time interval starting at 10 and ending anytime between 30 and 50. During this time, one unit of the Arm resource is used. The fluent

$$f_2 = (\mathrm{On}(\mathrm{dish1}, \mathrm{table1}), [[5, 5], [11, 11]], \emptyset, [40, 40][10, 10][46, 46][29, 29])$$

expresses the spatio-temporal fact that the dish is located at $[40, 40][10, 10][46, 46][29, 29]$ (in $\mathrm{table1}$'s reference frame) between time 5 and time 11.

Henceforth, we indicate with $(\cdot)^{(f)}$ an element of the five-tuple pertaining to fluent f. A pair of fluents (a, b) is *possibly concurrent* if $I^{(a)} \cap I^{(b)} \neq \emptyset$.

Fluents represent predicates with attached spatial, temporal and resource usage semantics. In this work, we present an automated decision making system which can reason about all these semantics in order to deduce temporally, spatially and resource feasible plans which achieve given goals. To do so, several reasoning algorithms are combined to obtain an overall system that can handle the different semantics of fluents. These include propositional planning, spatial and temporal reasoning, and scheduling techniques. All algorithms are constraint-based, and operate on a particular viewpoint [1] of fluents. All algorithms share a common constraint-based representation of the search space which represents the spatio-temporal evolution of the system:

Definition 2: A *constraint network* is a pair $(\mathcal{F}, \mathcal{C})$, where $\mathcal{F}$ is a set of *fluents* and $\mathcal{C}$ is a set of *constraints* among fluents in $\mathcal{F}$.

In the following, we introduce the types of constraints that can be represented in a constraint network.

A. Temporal Constraints

Fluents can be bound by *temporal constraints*. These are binary relations in the form $f_1 \ r \ f_2$ which restrict the relative placement in time of fluents f_1, f_2. Temporal constraints are relations in Allen's Interval Algebra (IA) [2], and restrict the possible bounds for the fluents' flexible temporal intervals $I^{(f_1)}$ and $I^{(f_2)}$. Atomic Allen's interval relations are the thirteen possible temporal relations between intervals, namely "before" (b), "meets" (m), "overlaps" (o), "during" (d), "starts" (s), "finishes" (f), their inverses (e.g., b^{-1}), and "equals" ($\equiv$). For example, the relation $f_2 \ \{\mathrm{m}\} \ f_1$ represents that f_2 ends as soon as f_1 starts.

Let the set of all thirteen Allen relations be B_{IA}. Each IA relation is a disjunction of atomic relations $\{r_1, \ldots, r_n\} \in B_{\mathrm{IA}} \times \ldots \times B_{\mathrm{IA}}$. In this work, we use *convex* IA relations [3].

Definition 3: A *temporal constraint network* is a pair $N = (V, TC)$, where

- $V = \{V_1, \ldots, V_n\}$ is a set of *variables* representing flexible temporal intervals;
- $TC : V \times V \rightarrow 2^{B_{\mathrm{IA}}}$ is a *mapping* which defines the binary constraints over the variables.

We say that a constraint network $M = (\mathcal{F}, \mathcal{C})$ *subsumes a temporal constraint network* $M_t = (V, TC)$, where $V = \{I^{(f)} : f \in \mathcal{F}\}$ and $TC = \{I^{(f_1)} \ r \ I^{(f_2)} : f_1 \ r \ f_2 \in \mathcal{C}\}$.

B. Spatial Constraints

As noted earlier, some fluents assert spatial properties on one or more spatial entities. Among predicates stating spatial semantics, we distinguish the predicate "On". This predicate reflects spatial knowledge on the placement of a spatial entity. We call a fluent whose predicate is "On" a *spatial fluent*.

In order to model spatial knowledge for use by the robot, we employ a new algebra called ARA$^+$ (Augmented Rectangle Algebra Plus) [4]. Constraints in ARA$^+$ bound the relative placement of spatial entities. ARA$^+$ allows to specify qualitative spatial knowledge, such as "the fork should be left of the dish". This high level of abstraction is essential for easily specifying how the robot should act to achieve specific spatial layouts. As opposed to other spatial calculi like Region Connection Calculus or Cardinal Algebra [5], ARA$^+$ also provides clear and useful metric semantics which are directly understandable by the robot. The algebra builds on a known, purely qualitative spatial algebra called Rectangle Algebra (RA) [6], which is an extension of IA to two dimensions. Variables in ARA$^+$ and in RA represent spatial entities of interest and are, respectively, bounded rectangles and rectangles whose sides are parallel to the axes of some orthogonal basis in a two-dimensional Euclidean space. ARA$^+$ and RA subsume both topology and cardinal relations.

The set B_{ARA^+} of atomic relations in ARA$^+$ is defined as $\{\langle r_x[l_1, u_1], \ldots, [l_n, u_n], r_y[l_1, u_1], \ldots, [l_m, u_m]\rangle : r_x, r_y \in B_{\mathrm{IA}}\}$. The IA relations r_x and r_y express qualitative one-dimensional relations between the projections A_x, B_x, A_y, B_y of rectangles A and B on the axes of the reference frame. The additional bounds augment the qualitative relations r_x and r_y with metric semantics. For example, the relation $B \ \langle \mathrm{b}[5, 13], \mathrm{b}\rangle \ A$ states that the horizontal distance between A and B should be at least 5 and at most 13; the vertical distance, on the other hand,

is a qualitative relation stating that "A is vertically higher than B". Also, the ARA$^+$ relation subsumes the qualitative relation "A is Northeast of B", as well as "A and B are disjoint" (see Figure 1). Overall, the given ARA$^+$ relation restricts the placements of A and B to those in which $A_x^- > B_x^+ + 5$ and $A_x^- < B_x^+ + 13$.

The set of ARA$^+$ relations is the power set of B_{ARA^+}. Each ARA$^+$ relation is a disjunction of atomic relations that model the possible mutual placement of two spatial entities.

Definition 4: A *spatial constraint network* is a pair $N = (V, SC)$, where

- $V = \{V_1, \ldots, V_n\}$ is a set of *variables* representing axis-parallel rectangles;
- $SC : V \times V \to 2^{B_{\text{ARA}^+}}$ is a *mapping* which defines the binary constraints over the variables.

We say that a constraint network $M = (\mathcal{F}, \mathcal{C})$ *subsumes a spatial constraint network* $M_s = (V, SC)$, where $V = \{br^{(f)} : f \in \mathcal{F}\}$ and $SC = \{br^{(f_1)}\ r\ br^{(f_2)} : f_1\ r\ f_2 \in \mathcal{C}\}$.

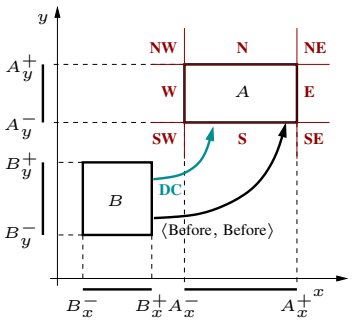


Fig. 1: Relation $B\ \langle \text{b}, \text{b} \rangle\ A$ in RA.

In addition to binary relations, ARA$^+$ includes unary relations to model size and perceived absolute placements of objects. The unary relation $\mathit{Size}[l_x, u_x][l_y, u_y]$ bounds the distances between two points of the same rectangle along one axis, constraining the minimum and maximum x and y dimensions to l_x, u_x, l_y and u_y. We also introduce the relation $\mathit{At}[l_x^1, u_x^1][l_y^1, u_y^1][l_x^2, u_x^2][l_y^2, u_y^2]$, which bounds the absolute placement of spatial entities. The bounds $[l_x^1, u_x^1][l_y^1, u_y^1]$ determine the position of the lower left corner (x^1, y^1), while $[l_x^2, u_x^2][l_y^2, u_y^2]$ determine the position of the upper right corner (x^2, y^2).

Since RA is subset of ARA$^+$ and is intractable [6], also ARA$^+$ with full expressiveness is intractable. Therefore, reasoning in full ARA$^+$ would be cumbersome if used on-line, especially in our application, where spatial reasoning is performed frequently. We thus focus on *convex* ARA$^+$ relations, which are relations whose qualitative components are convex RA relations. Convex RA relations impose convex disjunctions of IA relations on each axis. For example, $A\ \{\langle \text{b}, \text{o} \rangle, \langle \text{m}, \text{o} \rangle\}\ B$, is convex because the IA relation in the x dimension $A_x\ \{\text{b}, \text{m}\}\ B_x$ is convex, as is the elementary relation $A_y\ \{\text{o}\}\ B_y$. Note that specifying bounds on an otherwise convex qualitative relation is not always possible. For instance, $A_x\ \{\text{b}, \text{m}\}\ B_x$ is convex because its semantics can be seen as a single metric relation: $A_x^+ \leq B_x^-$. Conversely, $A_x\ \{\text{b}[5, \infty), \text{m}\}\ B_x$ is not convex, as expressing its metric semantics requires the disjunctive metric relation $A_x^+ + 5 < B_x^- \ \vee\ A_x^+ = B_x^-$. We thus disallow to specify bounds on ARA$^+$ relations composed of non-atomic relations. As shown in Lemma 1, this allows to reason about spatial relations in polynomial time. As we show in Section III, tractable temporal and spatial reasoning is essential for the realization of an efficient planner.

In general, the negation of a convex relation is not convex. Therefore imposing convexity also excludes the use of negation, e.g., it is not possible to model that the fork should "not" be right of the dish with ARA$^+$.

Convex ARA$^+$ relations are a powerful representational tool, which allow to model both detailed metric relations and high-level qualitative ones. For example, the relations

$$\texttt{Fork}\ \langle \text{d}[5, +\infty)[5, +\infty), \text{d}[5, +\infty)[5, +\infty) \rangle\ \texttt{Table} \quad (1)$$

$$\texttt{Fork}\ \langle \text{b}[10, 15], \text{d} \rangle\ \texttt{Dish} \quad (2)$$

$$\texttt{Fork}\ \mathit{Size}[2, 2][15, 15] \quad (3)$$

state that forks should be at least 5cm from the edge of the table (1), that they should be located to the left of dishes (2), and that the size of forks is 2×15 cm^2 (3).

The fundamental problem in reasoning about spatial relations is *consistency checking*:

Definition 5: A spatial constraint network $M_s = (V, SC)$ is said to be *consistent* if there exists an assignment of values to bounded rectangles in V such that all constraints in SC are satisfied.

Definition 6: Let $M = (\mathcal{F}, \mathcal{C})$ be a constraint network whose variables $\mathcal{F}$ are fluents and constraints $\mathcal{C}$ are temporal or ARA$^+$ relations. M is said to be *spatially consistent* if the spatial constraint network $M_s = (V, SC)$ subsumed by M is consistent.

As we will see, the spatial consistency of a constraint network can be used to determine that the spatial layout of objects observed by a robot is consistent with respect to a given, high-level spatial model. The following result is essential for guaranteeing that such operations can be performed on-line by a robot acting in the environment.

Lemma 1: Proving the consistency of a spatial constraint network $M_s = (V, SC)$ where all ARA$^+$ relations in SC are convex, is tractable.

Proof: Omitted — follows from a method introduced by van Beek [7] for translating qualitative IA relations to metric ones, as well as results by Condotta ([8], Theorem 2) and Balbiani [6]. □

Checking the spatial consistency of a constraint network with convex ARA$^+$ relations can be reduced to two Simple Temporal Problems (STP, [9]), one for each axis. Complete methods for checking the consistency of STPs run in low-order polynomial time [10]. If the network is consistent, the result of consistency checking is a set of admissible bounds on the placement of rectangles, a metric solution which is directly understandable by the robot.

C. Causal Domain Representation

Constraint networks as defined above can be used to represent the evolution of the state of a robot and of its

environment. In order to reason upon how this evolution can be changed by the robot, e.g., to achieve a particular goal, we require the notion of *planning operator*:

Definition 7: An *operator* is a pair $(f, (F, TC))$ where

- $f = (\mathrm{P}, \cdot, \cdot, u, \cdot)$ is a fluent representing an *action*;
- F_p is a set of *precondition fluents*, i.e., fluents that are required in order to execute the action;
- F_e is a set of *effect fluents*, i.e., fluents that are produced as a result of executing the operator;
- TC is a set of temporal constraints among fluents in $F_p \cup F_e \cup \{f\}$.

For example, the operator

$$\begin{aligned} f &= (\mathrm{Place}(?\mathrm{object}, ?\mathrm{location}), \cdot, \cdot, u(\mathrm{Arm}) = 1, \cdot) \\ F_p &= \{f_1 = (\mathrm{Hold}(?\mathrm{object}), \cdot, \cdot, u(\mathrm{Arm}) = 1, \cdot)\} \\ F_e &= \{f_2 = (\mathrm{On}(?\mathrm{object}, ?\mathrm{location}), \cdot, \cdot, \cdot, \cdot)\} \\ TC &= \{f \,\{\mathrm{m}^{-1}\}\, f_1, f \,\{\mathrm{s}, \mathrm{o}\}\, f_2\} \end{aligned}$$

describes the temporal, causal and resource-related aspects of the action of placing an object. The causal aspect expresses the fact that placing an object requires holding it, and results in the object being on the location. The temporal aspect is captured by the relations $\{\mathrm{m}^{-1}\}$ and $\{\mathrm{s}, \mathrm{o}\}$: the first expresses the fact that holding ceases to be true as soon as placing commences, while the latter that the object is on the location starting at the earliest when the action begins. The resource aspect is modeled by the usage of resource Arm, expressing the fact that placing requires one unit of this resource. Note that we have employed convex disjunctive temporal constraints to provide "loose" temporal coupling between the operator and its effect. This is convenient to take into account uncertainty in the execution of the behavior by the specific robot platform.

A fluent can be used to represent a goal, e.g.,

$$\begin{aligned} f_G = (&\mathrm{On}(\mathrm{cup1}, \mathrm{table1}), [[0, \infty), [0, \infty)], \emptyset, \\ &[0, \infty)[0, \infty)[0, \infty)[0, \infty)) \,, \end{aligned}$$

represents the fact that $\mathrm{cup1}$ should be on $\mathrm{table1}$ at an unspecified time in an unspecified position (as $I^{(f_G)}$ and $br^{(f_G)}$ are unbounded). An initial condition stating that the robot is holding $\mathrm{cup1}$ can be represented as

$$f_I = (\mathrm{Hold}(\mathrm{cup1}), [[10, 10], [11, \infty)], u(\mathrm{Arm}) = 1, \emptyset) \,.$$

The constraint network $M = (\{f_G, f_I\}, \emptyset)$ thus represents a desired, albeit under-specified, evolution in time of the system. The operator Place is applicable in this constraint network because its precondition $(\mathrm{Hold}(?\mathrm{object}), \cdot, \cdot, u(\mathrm{Arm}) = 1, \cdot)$ *unifies* with f_I. As a result of its instantiation, we obtain the following constraint network $(\mathcal{F}, \mathcal{C})$:

$$\begin{aligned} \mathcal{F} = &\{f_G, f_I, f' = (\mathrm{Place}(\mathrm{cup1}, \mathrm{table1}), [[11, 11][12, \infty)], \\ &u(\mathrm{Arm}) = 1, \emptyset)\}, \\ \mathcal{C} = &\{f_G \,\{\mathrm{o}^{-1}, \mathrm{s}^{-1}\}\, f', f' \,\{\mathrm{m}^{-1}\}\, f_I\} \end{aligned}$$

This constraint network is said to be *causally feasible*, as the goal can be reached from the initial constraint network. The network also happens to be temporally and spatially consistent, and the temporal bounds of fluent f' are a consequence of the constraint $f' \,\{\mathrm{m}^{-1}\}\, f_I$. Note that $br^{(f_G)}$ remains unbounded, as there are no spatial constraints affecting this fluent. However, suppose that the initial constraint network had also contained two more pieces of knowledge: the size of the table, namely $f'_I \; Size[100, 100][100, 100]$ where f'_I is a spatial fluent representing the table; and a requirement for the cup to be contained on the table, namely $f_G \; \langle \mathrm{d}, \mathrm{d} \rangle \; f'_I$. In this case, $br^{(f_G)}$ would have been refined through spatial consistency checking to $[1, 99][1, 99][1, 99][1, 99]$. Any rectangle within these bounds represents a feasible placement of the cup according to the spatial constraints in the network, and this rectangle can be directly used to extract the (x, y) coordinates of the arm's goal.

III. Reasoning

In order to be executable by a robot, a plan represented by a constraint network must be temporally consistent, as the temporal constraints imposed by the operators must be upheld. However, this condition is not sufficient. For example, a plan which requires the robot's arm to perform two tasks at the same time is not feasible. More in general, we define a *feasible plan* as follows.

Definition 8: A plan $(\mathcal{F}, \mathcal{C})$ is *feasible* iff it is

- temporally and spatially consistent;
- *resource feasible*, i.e., temporally overlapped fluents do not over-consume resources;
- *spatially feasible*, i.e., temporally overlapped spatial fluents are not spatially overlapped[1];
- *adherent to general spatial knowledge*, i.e., it remains spatially consistent in the presence of constraints modeling requirements on layout;
- *causally feasible*, i.e., it represents a plan which achieves given goals.

Our reasoning framework can be summarized as a collection of six solvers. Its inputs are a constraint network representing the initial condition, and a collection of spatial, temporal, causal and resource knowledge characterizing the robot's abilities and its environment:

Definition 9: A planning *domain* is a triple $\mathcal{D} = (\mathcal{O}, SK, \mathcal{RC})$ where $\mathcal{O}$ is a set of operators, SK is an ARA$^+$ constraint network describing requirements on the spatial layout of objects, and $\mathcal{RC}$ is a set of resource capacities.

Each of the six solvers employs the knowledge in the domain to enforce one of the criteria above on the initial network, thus transforming it into a feasible plan. Temporal and spatial consistency is enforced by low-order polynomial time constraint propagation methods, as mentioned in the previous sections. The remaining four solvers progressively transform the initial condition into a feasible plan by adding fluents and/or constraints until feasibility is established. The fluents and constraints added by each solver can be seen as *resolvers* of *conflicts*: conflicts are identified by checking for the presence of particular patterns of fluents and constraints in the common constraint network, and resolvers are constraints

[1] unless they have the properties of being container or supporter.

and/or fluents which eliminate the conflicting condition. An overall search algorithm is employed to search in the space of conflict resolvers (see Section IV).

A. Resource Feasibility

Resource feasibility is enforced through the so-called precedence constraint posting method. Conflicts are sets of fluents that are concurrent and over-consume one or more resources, i.e., $F \subseteq \mathcal{F}$ constitutes a conflict if

$$\exists R \in \mathcal{R} : \bigcap_{f \in F} I^{(f)} \neq \emptyset \ \wedge \ \sum_{f \in F} u^{(f)}(R) > \mathrm{Cap}(R). \quad (4)$$

Given that resource over-consumption is the result of two conditions, it is sufficient to remove one condition in order to resolve the conflict. In the approach used here, first described by Cesta et al. [11], resolvers are precedence constraints in the form $f_i \ \{b\} \ f_j$ which eliminate the temporal overlap between concurrent over-consuming fluents. Note that for every conflict there may be more than one possible sequencing. For instance, the pair of fluents

$$\begin{aligned} \{f_1 =& (\mathrm{Hold}(\mathrm{cup1}), [[5,5][20,20]], u(\mathrm{Arm}) = 1, \emptyset), \\ f_2 =& (\mathrm{Hold}(\mathrm{fork1}), [[5,10][50,50]], u(\mathrm{Arm}) = 1, \emptyset)\} \end{aligned}$$

are temporally overlapping and require a combined resource usage of 2 for the Arm resource in the interval $[5, 20]$. If the capacity of the arm is 1, these two fluents constitute a conflict with two possible resolvers, namely $f_1 \ \{b\} \ f_2$ or $f_2 \ \{b\} \ f_1$. By posting one of the resolvers to the common constraint network, the solver will enforce the sequencing of these two fluents, as well as the consequent shift in time of any other fluent which depends on them by means of temporal constraint propagation.

B. Spatial Feasibility

Enforcing spatial feasibility equates to disallowing the placement of objects on overlapping areas concurrently, namely all pairs of fluents $(f_i, f_j) \in \mathcal{F} \times \mathcal{F}$ such that

$$I^{(f_i)} \cap I^{(f_j)} \neq \emptyset \ \wedge \ br^{(f_i)} \cap br^{(f_j)} \neq \emptyset. \quad (5)$$

Each of these pairs is a conflict, whose resolvers are the two alternative temporal ordering constraints $f_i \ \{b\} \ f_j$ and $f_j \ \{b\} \ f_i$ that remove the temporal overlap.

C. Adherence to General Spatial Knowledge

General spatial knowledge SK prescribes a desired spatial layout, e.g.,

$$\text{Fork } \langle \mathrm{b}, \mathrm{d} \rangle \text{ Dish}, \qquad \text{Knife } \langle \mathrm{b}^{-1}, \mathrm{d} \rangle \text{ Dish}$$

(forks left of dishes, knives right of dishes). The following reflects the observation at time 10 of a fork and a knife:

$$\begin{aligned} f_{\mathrm{fork1}} =& (\mathrm{On}(\mathrm{fork1}, \mathrm{table1}), [[10,10][11,\infty)], \emptyset, br_1), \\ f_{\mathrm{knife1}} =& (\mathrm{On}(\mathrm{knife1}, \mathrm{table1}), [[10,10][11,\infty)], \emptyset, br_2), \\ & br_1 \ At[40,40][10,10][46,46][29,29], \\ & br_2 \ At[31,31][11,11][37,37][30,30]. \end{aligned}$$

These observations are inconsistent with SK, as the positions of fork and knife are specular to the desired layout. This situation constitutes a conflict. Formally, given $F \subseteq \mathcal{F}$, let $\mathcal{C}|_F$ be the constraints in $\mathcal{C}$ involving fluents in F. F constitutes a conflict if

$$\bigcap_{f \in F} I^{(f)} \neq \emptyset \ \wedge \ (F, \mathcal{C}|_F \cup SK) \text{ is spatially inconsistent.} \quad (6)$$

The source(s) of inconsistency are the *At* constraints deriving from observation. The conflict in the example above can be resolved by inverting the positions of fork and knife. Computing new positions for the two objects is done by removing the two *At* constraints from $(F, \mathcal{C}|_F \cup SK)$ and computing the bounding boxes of the fluents f_{fork1} and f_{knife1}. The bounded rectangles resulting from spatial constraint propagation in this network, br'_1 and br'_2, suggest new placements for the fork and knife which are consistent with respect to SK. The resolver of this conflict is a set of fluents which represents these new placements as goals, namely

$$\begin{aligned} f'_{\mathrm{fork1}} =& (\mathrm{On}(\mathrm{fork1}, \mathrm{table1}), [[0,\infty)[0,\infty)], \emptyset, br'_1) \\ f'_{\mathrm{knife1}} =& (\mathrm{On}(\mathrm{knife1}, \mathrm{table1}), [[0,\infty)[0,\infty)], \emptyset, br'_2) \end{aligned}$$

The resolver also includes temporal constraints modeling the fact that the new goals should be achieved after the observed fluents: $f_{\mathrm{fork1}} \ \{\mathrm{b}\} \ f'_{\mathrm{fork1}}$ and $f_{\mathrm{knife1}} \ \{\mathrm{b}\} \ f'_{\mathrm{knife1}}$. The goal fluents in the common constraint network are seen as conflicts by the causal feasibility solver (see below), which in turn will propose resolvers achieving these goals. These resolvers will be ground operators for picking and re-placing the two objects into their new positions. Finally, the resolver also includes the constraints in SK, as their presence in the plan constrains the bounding boxes of all spatial fluents to adhere to the required layout.

In the general case, conflicts in adherence to spatial knowledge may have many resolvers. In our example, it would have been alternatively possible to re-place the fork to the left of the knife, rather than re-placing both objects; yet another possibility would have been to pick and place the knife only. The choice of an appropriate resolver is guided by heuristics employed by the overall search process which we describe in the following section.

D. Causal Feasibility

This solver employs a description of operators defined as shown in Section II-C. Conflicts for this solver are goal fluents (or sub-goals posted by the solver which enforces adherence to general spatial knowledge), and resolvers are ground operators whose effects are the (sub-)goals. The use of this solver within the overall search realizes a goal-regression planner [12].

IV. Search and Solution Extraction

The conflicts identified by each solver constitute *decision variables* in a high-level Constraint Satisfaction Problem (CSP). A conflict is resolved by adding one of its possible resolvers to the common constraint network. When all conflicts are resolved, the constraint network represents a feasible plan. Choosing one resolver for a conflict identified by one solver may not be consistent with the choice of another resolver for another conflict. For instance, a particular ordering of tasks due to over-use of the Arm resource may not be possible due to an ordering previously chosen for enforcing spatial feasibility. This "cross-validation" is made possible by

the common constraint network, which is used to propagate the spatial, temporal, resource and causal consequences of all resolvers. For this reason, it is necessary to search in the space of possible resolvers.

Given a conflict d identified by a solver, we denote its possible resolvers as the set of constraint networks $\delta^d = \{(F_r^d, C_r^d)_1, \ldots, (F_r^d, C_r^d)_n\}$.

Function `Backtrack` $(\mathcal{F}_I, \mathcal{C}_I)$: success or failure

1 $d \leftarrow$ `Choose` $((\mathcal{F}_I, \mathcal{C}_I), h_{conf})$
2 **if** $d \neq \emptyset$ **then**
3 $\delta^d = \{(F_r^d, C_r^d)_1, \ldots, (F_r^d, C_r^d)_n\}$
4 **while** $\delta^d \neq \emptyset$ **do**
5 $(F_r^d, C_r^d)_i \leftarrow$ `Choose` (d, h_{res})
6 **if** $(\mathcal{F}_I \cup F_r^d, \mathcal{C}_I \cup C_r^d)$ *is temporally and spatially consistent* **then**
7 **return** `Backtrack` $(\mathcal{F}_I \cup F_r^d, \mathcal{C}_I \cup C_r^d)$
8 $\delta^d \leftarrow \delta^d \setminus \{(F_r^d, C_r^d)_i\}$
9 **return** *failure*
10 **return** *success*

Given an initial constraint network $(\mathcal{F}_I, \mathcal{C}_I)$ containing one or more goal fluents, Algorithm `Backtrack` searches for a set of resolvers to be added to $(\mathcal{F}_I, \mathcal{C}_I)$ in order to impose feasibility. The algorithm is a systematic CSP-style backtracking search. Conflicts to branch on are chosen according to a conflict ordering heuristic h_{conf} (line 1); the alternative resolving constraint networks are chosen according to a resolver ordering heuristic h_{res} (line 5). The former decides which conflict to attempt to resolve first, e.g., to re-place an object or to free the arm, while the latter decides which resolver to attempt first, e.g., which object has to be replaced. Note that adding resolving constraint networks may entail the presence of new conflicts to be resolved, e.g., pick and place actions which solve general spatial knowledge infeasibility can lead to new resource conflicts, as the newly generated actions must be scheduled with the already existing actions.

The heuristic used for resolver selection, h_{res}, depends on the type of conflict. Resolvers of resource and spatial conflicts are ordered according to a known heuristic which prefers orderings that maximize temporal flexibility [11] (outside the scope of this paper). Resolvers that impose adherence to general spatial knowledge are ordered taking into account that they entail robot manipulation. The heuristic favors resolvers that involve few objects, the rationale being that moving fewer objects is less prone to failure than moving many. Also, the heuristic favors moves which least affect the *spatial flexibility* of the resulting placement. This is a function $\text{Flex}(\mathcal{F}, \mathcal{C} \setminus c)$, where c is the set of observed *At* constraints on objects to be re-placed. It builds on the notion of rigidity [4] to compute the "spatial slack" achievable by applying a resolver: high flexibility entails that objects will be re-placed into positions which are farther from failure, therefore affording less precise manipulation.

A. Spatial Solution Extraction

Algorithm `Backtrack` enforces, among other requirements, that all the bounded rectangles of spatial fluents adhere to general spatial knowledge. Among the many feasible placements subsumed by the bounded rectangle of a spatial fluent, one must be chosen for execution. In a robotic context, we are interested in obtaining the placement that has maximum distance from the lower and upper bounds in both dimensions, as the region that is given to a robot to place an object should tolerate the inaccuracy of manipulation. In other words, if the robot does not place an object exactly within the region, the spatial layout should still be consistent. For this reason, we prefer assignments that are close to the center of the solution space.

Given $M_s = (V, SC)$, we compute an approximation of the most centered solution for each bounded rectangle $A \in V$ by leveraging the concept of 2D representation of an interval [13]. The interval A_x (similarly for A_y) is represented as a window in the space of start and end position (see Figure 2). The window is characterized by four numbers, namely, minimum and maximum positions, and minimum and maximum lengths. All possible placements of A_x after spatial consistency is enforced are within a convex polygon in the 2D space. We choose as "most centered" placement for A_x the center of mass of this polygon, thus obtaining an assignment of A_x^+ and A_x^-.

The choice of most centered placement for one object affects, through the spatial constraints in the network, the possible choices for other objects. To compute placements for all objects, we thus (a) extract the center of mass for the x and y intervals of one bounded rectangle, (b) add one *At* constraint reflecting this choice, and (c) apply spatial consistency to update the bounds of the other rectangles. The computational overhead of this procedure is $\Theta(|V|^3)$.

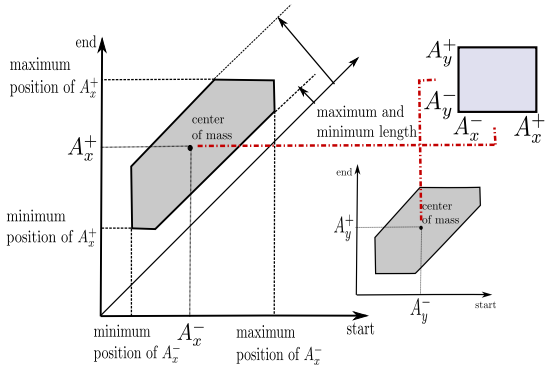


Fig. 2: Representation of a rectangle in two 2D intervals.

Overall, the plans obtained as a result of backtracking, solution extraction and constraint propagation are guaranteed to adhere to all modeled knowledge. Due to the choice of complete algorithms for each consistency and feasibility enforcing strategy, the system is guaranteed to find a solution if one exists. More formally,

Theorem 1: Algorithm `Backtrack` is complete.

Proof: The temporal consistency of the common constraint network is enforced through a path consistency algorithm [10], which is complete for convex temporal constraint networks [9]. Consistency checking of convex ARA$^+$ relations can be reduced to the former problem as well (see Section II-B). Causal feasibility is enforced with backwards search, which is also complete [12]. Resource, spatial and

(a) (b) (c) (d)

Fig. 3: Snapshots of the first experiment with a physical robot — initial situation (a); general spatial knowledge vs. observed placements (b); execution of a pick action (c); achieved placements (d).

general knowledge feasibility solvers enumerate all possible conflicts (see eqs. (4) to (6)) and all their resolvers. Combined with the completeness of backtracking search [14] (Algorithm `Backtrack`), this guarantees that either a plan is found or all states in the search space are proved infeasible. □

V. Experiments

Five experiments were carried out to validate the feasibility of our approach[2]. The first three involve the use of a MetraLabs G5 robot base equipped with a Kinova Jaco arm and an Asus XtionPro RGB-D camera for 3D perception. In order to facilitate manipulation, a cup was used instead of a dish, and forks and knifes were equipped with graspable appendices. In the first run, the robot was to achieve the goal of placing the cup on the table. The following domain was given to the robot:

Causal & temporal knowledge $\mathcal{O}$:
(operators for sensing, picking and placing, omitted)
Resource knowledge $\mathcal{RC}$: $\mathrm{Cap}(\mathrm{Arm}) = 1$
General spatial knowledge SK:
`Fork` $\langle \mathrm{d}[5,+\infty)[5,+\infty), \mathrm{d}[5,20][5,+\infty)\rangle$ `Table`
`Knife` $\langle \mathrm{d}[5,+\infty)[5,+\infty), \mathrm{d}[5,20][5,+\infty)\rangle$ `Table`
`Cup` $\langle \mathrm{d}[5,+\infty)[5,+\infty), \mathrm{d}[5,20][5,+\infty)\rangle$ `Table`
`Fork` $\langle \mathrm{b}[10,15], \mathrm{d}^{-1}\rangle$ `Cup`, `Knife` $\langle \mathrm{b}^{-1}[10,15], \mathrm{d}^{-1}\rangle$ `Cup`
`Table` $Size[58,58][58,58]$, `Knife` $Size[4,5][15,18]$,
`Fork` $Size[4,5][15,18]$, `Cup` $Size[5,7][5,7]$

In addition to specifying the minimum distance of 5cm from the edge of the table for all objects, the general spatial knowledge imposes that they should be at most 20cm from the side of the table from which the robot is operating due to reachability constraints.

In the initial state, the robot holds the cup. When the goal is posted, a plan consisting of the actions $\mathrm{Sense}(\mathrm{table})$ and $\mathrm{Place}(\mathrm{cup}, \mathrm{table})$ is generated. When dispatched, the $\mathrm{Sense}(\mathrm{table})$ action activates the 3D-perception module, which generates spatial fluents representing the perceived locations of fork and knife. In this experiment, the fork and knife were placed too close to each other (see Figure 3(a)), thus the plan turns out to be infeasible due to lack of adherence to general spatial knowledge. The amended plan resulting from the application of Algorithm `Backtrack` includes placing the cup on the tray to free the arm, moving the fork to the left according to the new placements extracted by the "most centered solution", placing the cup on the table, and moving the knife to the right. Notice that the decision to employ the tray was due to resource conflict and spatial conflict, and the order of re-placements was derived automatically by the spatial feasibility resolver.

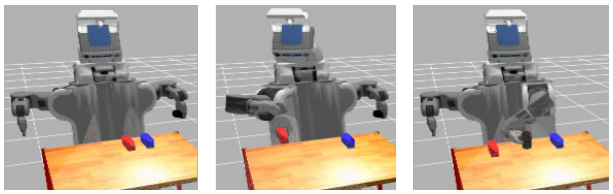
In the second experiment, the initial goal and knowledge was the same, but the fork and knife were swapped, and were far enough from each other to accommodate the cup between them. The algorithm generated a plan in which the cup is first placed on the table, and then the tray is used to swap fork and knife.

The third experiment demonstrates the use of another specification of general spatial knowledge in ARA$^+$, namely a layout such that the fork and the knife should both be on the left of the cup. The model is identical to the first specification with the exception of the relation between knife and cup, which becomes `Knife` $\langle \mathrm{b}[8,10], \mathrm{d}\rangle$ `Cup`.

As a demonstration of how our model can be changed to suit different physical capabilities and environments, we ran two experiments with a PR2 robot operating in a fully physically simulated environment (Gazebo, see Figure 4). The specification was changed only to reflect: (1) the size of the objects, (2) the capacity of the Arm resource (the PR2 has two arms), and (3) the addition of a $\mathrm{Move}(?\mathrm{from}, ?\mathrm{to})$ operator. The environment contained two tables, $\mathrm{table1}$ and $\mathrm{table2}$. The PR2 was given the goal to place a cup on $\mathrm{table2}$, which in the initial situation was on $\mathrm{table1}$. The obtained plan was: $\mathrm{Sense}(\mathrm{table1})$, $\mathrm{Pick}(\mathrm{cup}, \mathrm{table1})$, $\mathrm{Move}(\mathrm{table1}, \mathrm{table2})$, $\mathrm{Sense}(\mathrm{table2})$, $\mathrm{Place}(\mathrm{cup}, \mathrm{table2})$. When $\mathrm{table2}$ is reached, new observations reveal that the situation does not adhere to general spatial knowledge. Thanks to the higher capacity of the Arm resource, the new plan does not require to use a tray to free an arm for further manipulation. The fifth experiment is a variant of the fourth, where the spatial flexibility heuristic leads the robot to replace the knife rather than the fork.

Although the causal subproblem (task planning) is PSPACE-hard [12], employing tractable reasoners for the frequent consistency checking involved results in a practically usable system (in all experiments, the plan is found in less than three seconds).

[2]Videos available at `http://aass.oru.se/~mmi/ICRA-2014`.



(a) (b) (c)

Fig. 4: A PR2 employing spatial knowledge to set a table.

A. Discussion and Conclusions

The problem of combining qualitative spatial knowledge with perception, planning and actuation has been addressed in perceptual anchoring [15] and cognitive vision [16]. Qualitative spatial reasoning has been used with robotic platforms for robot navigation and self-localization [17], motion planning [18] and task planning [19]. Some recent work has studied how to combine qualitative and metric spatial reasoning in robotics [20]. A more significant body of work has addressed how to endow robots with metric (as opposed to qualitative) spatial reasoning capabilities. Many focus on geometric reasoning, some employing metric constraints in combination with planning [21], [22] others proposing ad-hoc metric spatial reasoning for analyzing perceived context [23]. All of the above disregard the issue of combining qualitative and metric knowledge, nor do they allow to include resource and temporal reasoning.

Metric temporal reasoning has been recognized as an important dimensions of robot task planning [24], as have qualitative temporal models [25] and combined qualitative and metric temporal reasoning [26]. The latter type of models have been combined also with resource reasoning [27] for multi-robot systems. These approaches point to key advantages of using constraint-based representations, as they provide a common language for integrated reasoning. Our approach builds on this principle in order to integrate spatial reasoning into the planning framework. In addition, the modularity of our approach[3] facilitates the inclusion of further solving capabilities. In future work we will leverage this capability to include kinematic feasibility checking and 3D motion planning into our system.

Acknowledgments. This work is supported by the EC 7th Framework Program under project RACE (grant #287752).

References

[1] B. Smith, "Modelling, chapter 11," in *Handbook of Constraint Programming*, F. Rossi, P. van Beek, , and T. Walsh, Eds. Elsevier, 2006, pp. 377–406.

[2] J. Allen, "Towards a general theory of action and time," *Artif. Intell.*, vol. 23, no. 2, pp. 123–154, 1984.

[3] G. Ligozat, "A new proof of tractability for ORD-Horn relations," in *AAAI Workshop on Spatial and Temporal Reasoning*, 1996.

[4] M. Mansouri and F. Pecora, "A representation for spatial reasoning in robotic planning," in *Proc. of the IROS Workshop on AI-based Robotics*, 2013.

[5] J. Renz and B. Nebel, "Qualitative spatial reasoning using constraint calculi," in *Handbook of Spatial Logics*, 2007, pp. 161–215.

[6] P. Balbiani, J.-F. Condotta, and L. F. Del Cerro, "A new tractable subclass of the rectangle algebra," in *Proc. of the 16th Int'l Joint Conf. on Artif. Intell.*, vol. 1, 1999, pp. 442–447.

[7] P. van Beek, "Exact and approximate reasoning about qualitative temporal relations," Ph.D. dissertation, University of Waterloo, Waterloo, Ont., Canada, Canada, 1990, uMI Order No. GAXNN-61098.

[8] J.-F. Condotta, "The augmented interval and rectangle networks," in *Proc. of 7th Int'l Conf. on Principles of Knowledge Representation and Reasoning*, 2000.

[9] R. Dechter, I. Meiri, and J. Pearl, "Temporal constraint networks." *Artif. Intell.*, vol. 49, no. 1-3, pp. 61–95, 1991.

[10] L. Xu and B. Choueiry, "A new efficient algorithm for solving the simple temporal problem," in *Proc. of the 4th Int'l Conf. on Temporal Logic*, 2003.

[11] A. Cesta, A. Oddi, and S. F. Smith, "A constraint-based method for project scheduling with time windows," *Journal of Heuristics*, vol. 8, no. 1, pp. 109–136, January 2002.

[12] M. Ghallab, D. Nau, and P. Traverso, *Automated Planning: Theory and Practice*. Morgan Kaufmann, 2004.

[13] J.-F. Rit, "Propagating temporal constraints for scheduling," in *Proc. of the 2nd Nat'l Conf. on Artif. Intell.*, 1986.

[14] P. van Beek, "Backtracking search algorithms, chapter 4," in *Handbook of Constraint Programming*, F. Rossi, P. van Beek, , and T. Walsh, Eds. Elsevier, 2006, pp. 85–134.

[15] A. Loutfi, S. Coradeschi, M. Daoutis, and J. Melchert, "Using knowledge representation for perceptual anchoring in a robotic system," *International Journal on Artificial Intelligence Tools*, vol. 17, no. 5, pp. 925–944, 2008.

[16] W. Kennedy, M. Bugajska, M. Marge, W. Adams, B. Fransen, D. Perzanowski, A. Schultz, and J. Trafton, "Spatial representation and reasoning for human-robot collaboration," in *Proc. of the 22nd Nat. Conf. on Artif. Intell.*, 2007.

[17] D. Wolter and J. Wallgrün, *Qualitative spatial reasoning for applications: New challenges and the SparQ toolbox*. IGI Global, 2010.

[18] M. Westphal, C. Dornhege, S. Wölfl, M. Gissler, and B. Nebel, "Guiding the generation of manipulation plans by qualitative spatial reasoning," *Spatial Cognition and Computation: An Interdisciplinary Journal*, vol. 11, no. 1, pp. 75–102, 2011.

[19] L. Belouaer, M. Bouzid, and A. Mouaddib, "Ontology based spatial planning for human-robot interaction," in *Proc of the 17th Int'l Symp. on Temporal Representation and Reasoning*, 2010.

[20] L. Mosenlechner and M. Beetz, "Parameterizing actions to have the appropriate effects," in *Proc. of IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems*, 2011.

[21] F. Lagriffoul, D. Dimitrov, A. Saffiotti, and L. Karlsson, "Constraint propagation on interval bounds for dealing with geometric backtracking," in *Proc. of IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems*, 2012.

[22] J. Guitton and J.-L. Farges, "Taking into account geometric constraints for task-oriented motion planning," in *Workshop on Bridging the Gap between Task and Motion Planning (BTAMP)*, 2009.

[23] H.-Y. Jang, H. Moradi, S. Hong, S. Lee, and J. Han, "Spatial reasoning for real-time robotic manipulation," in *Proc. of IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems*, 2006.

[24] B. Williams, M. Ingham, S. Chung, and P. Elliott, "Model-based programming of intelligent embedded systems and robotic space explorers," *Proc. of the IEEE*, vol. 91, no. 1, pp. 212–237, 2003.

[25] P. Doherty, J. Kvarnström, and F. Heintz, "A temproal logic-based planning and execution monitoring framework for unmanned aircraft systems," *Journal of Automated Agents and Multi-Agent Systems*, vol. 2, no. 2, 2010.

[26] J. Bresina, A. Jónsson, P. Morris, and K. Rajan, "Activity planning for the mars exploration rovers," in *Proc. of the 15th Int'l Conf. on Automated Planning and Scheduling (ICAPS)*, 2005.

[27] M. D. Rocco, F. Pecora, and A. Saffiotti, "When robots are late: Configuration planning for multiple robots with dynamic goals," in *Proc. of IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems*, 2013.

[3]The implementation of our approach builds on the Meta-CSP Framework, an API for combining multiple reasoners, see `metacsp.org`.