Grandpa Hates Robots — Interaction Constraints for Planning in Inhabited Environments

Uwe Köckemann and **Federico Pecora** and **Lars Karlsson**
Center for Applied Autonomous Sensor Systems, Örebro University, Sweden
{uwe.koeckemann,federico.pecora,lars.karlsson}@oru.se

Abstract

Consider a family whose home is equipped with several service robots. The actions planned for the robots must adhere to *Interaction Constraints (ICs)* relating them to human activities and preferences. These constraints must be sufficiently expressive to model both temporal and logical dependencies among robot actions and human behavior, and must accommodate incomplete information regarding human activities. In this paper we introduce an approach for automatically generating plans that are conformant wrt. given ICs and partially specified human activities. The approach allows to separate causal reasoning about actions from reasoning about ICs, and we illustrate the computational advantage this brings with experiments on a large-scale (semi-)realistic household domain with hundreds of human activities and several robots.

We address the challenge of automatically generating plans that have to accommodate scheduled activities, features and preferences of a set of uncontrollable agents (such as humans). Consider, e.g., a domain in which a set of robots have to plan for a whole day in a household environment that is co-inhabited by a human family. The schedules and preferences of humans impose complex constraints on robot plans. We may, for instance, be required to avoid that a robot is vacuuming a room in which an inhabitant is reading. This could be resolved by separating these two events in time. The applicability of such constraints may not only be based on the activity (such as reading), but also on the preferences of the inhabitant. The above example may only apply to inhabitants that are easily distracted. In many cases we can also allow to violate these constraints by paying a social cost (Alami et al., 2006).

To handle such domains we propose to use *Interaction Constraints (ICs)* to model how robot plans and human activities should relate to each other. The building blocks of ICs are also constraints, albeit at a lower level of abstraction, which enforce temporal and logical requirements or costs on robot plans. We show through the use of examples how the expressiveness of constraint-based planning is well suited for modeling interaction requirements with humans. The constraint-based nature of the domain definition language also allows to account for partially specified human behavior. We propose a planning algorithm which leverages the richness of the language to decompose the overall problem into easier sub-problems that are separated from causal reasoning (assuming robot and human activities are only connected via ICs). The planner computes plans that are conformant wrt. given ICs and partially specified human activities. Finally, we exemplify how using off-the-shelf planning technology (i.e., modeling ICs as actions) would create a far more difficult search problem in the causal reasoner.

Related Research

In order to accommodate the behavior of uncontrollable agents automated planning must take into account some form of temporal knowledge about their activities. Also these activities may be only partially specified. One possibility is to model uncontrollable agent behavior as future events modeled as timed-initial literals. These can be either compiled into dummy actions (Coles et al., 2008; Benton, Coles, and Coles, 2012) or processed in an event queue (Do and Kambhampati, 2003). Both approaches imply a tight coupling between future events and plan search. Our approach exploits the fact that we can decouple future events from plan search under certain assumptions. Also these approaches are not adequate for partially specified behavior of uncontrollable agents. The same holds for timeline-based planning approaches (Chien et al., 2000; Bresina et al., 2005; Fratini, Pecora, and Cesta, 2008), in which temporal constraints can be used to model external events. Our work extends this idea to cater for partially specified events and other constraint types.

Conditional temporal plans can be used to describe alternative courses of action depending on the behavior of uncontrollable agents. These plans constitute temporal problems with choices. Tsamardinos, Vidal, and Pollack (2003) propose an approach to solve these problems, but do not focus on obtaining the plans. Mausam and Weld (2008) extend Markov Decision Processes to temporal planning. Planning with temporal and resource uncertainty is addressed by Beaudry, Kabanza, and Michaud (2010), who tackle the problem by attaching probability densities to durations and resource usages. The type of uncertainty addressed in these papers is different (and more difficult) than what we are in-

terested in here, namely un-/under-specified human activities.

Blythe (1994) studies the problem of planning and plan repair with probabilistic external events. This approach, however, only considers failures due to external events, and lacks ways to model and take into account preferences during plan generation.

Several approaches tackle the problem of human-aware planning explicitly. These include work by Cirillo, Karlsson, and Saffiotti (2009), who propose to handle human activities with a conditional planning approach (Karlsson, 2001). We borrow their notion of ICs, but reformulate it to account for qualitative and metric time, logical constraints and social costs. In addition, the conditional nature of their approach does not allow to decouple human activities from the search space of causal reasoning which negatively affects scalability. The work on human-aware task planning by Alami et al. (2006) and Montreuil et al. (2007) is based on hierarchical task networks and incorporates time, as well as social costs depending on human and robot actions. Their notion of social constraints only imposes costs and there seems to be no way to handle uncertainty about human actions. Conversely it has the advantage of allowing cooperation between robots and humans, an issue which we do not address in this paper.

Running Example

We now introduce an example domain for planning in inhabited environments. This domain models a house with a set of inhabitants that includes two parents, four kids, two infants and four grandparents. The house consists of 20 rooms with specific purposes (such as bedrooms, bathrooms, laundry room etc.).

The temporal horizon of the problems in this domain is 1440 minutes (24 hours). This means that a plan has to take place between 0 and 1440 minutes (where 0 represents 6:00am). Human activities are pre-scheduled and linked to locations in the house and there may be cases in which the human activity is unknown at planning time. The full set of activities we used here is *idle, out, eating, sleeping, reading, working, cooking, playing* and *usingBathroom*.

There are four robots, each with different capabilities, that have to perform a set of tasks. The list of tasks we use is *vacuum, sortStuff, assist, collectTrash, takeOutTrash, collectLaundry, doLaundry, cleanRoom, entertain*. Robots can either move from one location to another adjacent location or perform a task at their current location. To make the problems more interesting, we use temporal constraints on goals for a given day to divide them into three batches, where every goal in the first batch has to be achieved before every goal in the second batch. The goals in the second batch have to be achieved after all those in the first batch and before the ones in the third batch.

Robot plans have to follow certain rules to be executed alongside human schedules. If one of the inhabitants is using the bathroom, for instance, no robot is allowed to be in the same room at the same time. Some of these rules are based on preferences or features of the inhabitants. Here is a list of possible rules (rules marked with * can be violated by paying social cost): (1) No working where someone is sleeping*; (2) No working where a "light sleeper" is sleeping; (3) Don't interrupt easily distracted people*; (4) Robots not allowed when bathroom is occupied; (5) Robots avoid locations with people that dislike them*; (6) Collecting trash not allowed while people are eating*; (7) Robots not allowed in bedroom with two married people*; (8) No working in the kid's rooms while a kid is playing*; (9) Robots not allowed to vacuum while someone is reading*.

Constraint-based Planning

In this section we introduce a constraint-based formulation of the planning problem (Köckemann, Pecora, and Karlsson, 2014) and extend it to account for constraints between robot actions and partially specified activities of uncontrollable agents. The language we use describes states (or context), actions, goals and human activities via statements and constraints. Statements attach information to temporal intervals. Constraints are used to limit the allowed assignment of variables (e.g. an interval's duration). A context is described by a set of statements $\mathcal{F}$ and a set of constraints $\mathcal{C}$. We refer to the pair $\Phi = (\mathcal{F}, \mathcal{C})$ as a *Constraint Database (CDB)*. Let us start with an example CDB:

```
initial-context:
 statements:
  (a1,  activity(father,kitchen,cooking))
  (a2,  activity(mother,study,A))
  (/s, at(r1),robotRoom)
  (/s, at(r2),robotRoom)
  (/s, state(t1,clean,kitchen),waiting)
  (/s, state(t2,vacuum,study),waiting)
 constraints:
  a1 At [0,0] [59,59];
  a2 At [60,60] [106,106];
  A in {reading,working};
  s Release [0,0];
  _planningInterval(0,1440);
  socialCost() <= 100.0;
```

This initial context contains six statements: two human activities, two robot locations and two tasks ($t1$ and $t2$) in their initial state. Temporal intervals of these statements are $a1$, $a2$ and $/s$[1]. Human activities are statements that have a fixed interval (imposed by the *At* constraint, which sets earliest/latest start/end time). All other statements (about robot locations and task states) are constrained to start at 0 ($sRelease[0, 0]$). The second human activity A (interval $a2$) is unknown but has two possible values $A \in \{reading, working\}$. In this way we introduce uncertainty about the activity. The planning interval is set for 24 hours and the maximum social cost we allow is 100.0. In the experiments described later we will have four robots, between 110 and 260 human activities and between 10 and 30 tasks for each problem.

We will now introduce statements, constraints, actions, planning problem and solution. We use two types of *terms*: *Constant (or ground) terms* refer to specific objects, while *variable terms* refer to an object of a specific type. The human activity *(a2, activity(mother,study,A))*

[1] */s* is a syntactic element representing a set of intervals. This allows to express temporal constraints for multiple intervals.

above has two ground terms *mother* and *study* and one variable A. A *statement* is a tuple $(\mathcal{I}, x, v)$ where $\mathcal{I} = [[EST, LST], [EET, LET]]$ is a flexible temporal interval with Earliest and Latest Start and End Times (EST, LST, EET and LET) during which the statement holds. The domain of all time points EST, LST, EET and LET is implicitly set to $D_{time} = \{0, 1, 2, \ldots, T_{max}\}$, where $T_{max} \in \mathbb{N}_0$ is also called the temporal horizon of the planning problem (set with *planningInterval(0,1440)* in the example). The variable x has the form $p(t_1, \ldots, t_n)$, where p is the name of the variable and all t_i are terms. The term v is the assigned value (omitted for value $True$ as, e.g., in *activity* above). For every variable term t in a CDB we assume that it has a finite domain $D[t]$ of possible values. A substitution $\sigma = \{x_1/v_1, \ldots, x_n/v_n\}$ can be used on any structure (such as CDBs, statements, operators) that we use in this paper and will replace all occurrences of terms x_i in that structure with terms v_i. In the same way, we use $Ground(\Phi)$ to indicate that all terms that occur in CDB Φ are ground. The union of two CDBs Φ_1 and Φ_2 is $\Phi_1 \cup \Phi_2 = (\mathcal{F}_{\Phi_1} \cup \mathcal{F}_{\Phi_2}, \mathcal{C}_{\Phi_1} \cup \mathcal{C}_{\Phi_2})$. If $\mathcal{F}_1$ and $\mathcal{F}_2$ are sets of statements, let $\theta(\mathcal{F}_2, \mathcal{F}_1)$ be a set of substitutions s.t. $\sigma(\mathcal{F}_2) \subseteq \mathcal{F}_1, \forall \sigma \in \theta(\mathcal{F}_2, \mathcal{F}_1)$ (i.e., θ represents all ways to substitute $\mathcal{F}_2$ to be a subset of $\mathcal{F}_1$).

Constraints limit the allowed values for all variables in a CDB. Temporal constraints limit the earliest/latest start/end times of a flexible interval $\mathcal{I}$. Logical constraints limit allowed combinations of variable assignments. Formally, every constraint can be written down in a relational form $r(x_1, \ldots, x_n)$, where r is a name and x_i are terms. Let $D[x_i]$ be the domain of each variable. We assume the existence of a consistency function

$$Satisfied_{Type(r)} : r \times D[x_1] \times \ldots \times D[x_n] \rightarrow \{True, False\}$$

for each type of constraint ($Type(r)$) that decides whether or not c is satisfied given a (partial) assignment of all its variable terms. We use $Satisfied_{Type(c)}(c, \Phi)$ to express the same for any constraint c given a CDB Φ (skipping the relational form). A CDB Φ is *consistent* iff

$$Consistent(\Phi) \equiv \exists \sigma : Ground(\sigma(\Phi)) \\ \wedge \forall c \in \sigma(\mathcal{C}_\Phi) : Satisfied_{Type(c)}(c, \Phi).$$

Given this definition of $Consistent$, we need to provide a definition for $Satisfied$ for each type of constraint. For temporal constraints ($Satisfied_T$) we use quantitative Allen's interval algebra (Meiri, 1996). Allen's interval constraints (Allen, 1984) describe 13 qualitative relations between temporal intervals (such as a *Before/After/Equals/Overlaps* b). Quantitative Allen's relations extend this notion by attaching bounds. So, a *Before* b $[1, 10]$ means that a ends between 1 and 10 time-units before b starts. We also added unary constraints (such as *Release* and *At* from the example) and conceptually neighboring constraints (Freksa, 1992) (such as *BeforeOrMeets*). In addition to temporal constraints, we also include reusable resources ($Satisfied_R$), which are interpreted as a meta-CSP (Cesta, Oddi, and Smith, 2002) over $Satisfied_T$ (i.e., reasoning about resources includes temporal reasoning). Logical constraints ($Satisfied_L$) are evaluated as Prolog queries given background knowledge (Bratko, 2000) that is provided together with domain and/or problem description. Cost constraints ($Satisfied_C$) boil down to simple function evaluations and inequalities.

A CDB Φ_1 *supports* a second CDB Φ_2 iff

$$Supports(\Phi_1, \Phi_2) \equiv \exists \sigma \in \theta(\mathcal{F}_{\Phi_2}, \mathcal{F}_{\Phi_1}) : \\ Consistent(\Phi_1 \cup \sigma(\Phi_2))$$

requiring at least one way to make $\mathcal{F}_{\Phi_2}$ a subset of $\mathcal{F}_{\Phi_1}$ that is consistent with the constraints in both Φ_1 and Φ_2. This will be used later to decide if a set of goals has been reached by a plan or if the condition of an interaction constraint applies.

Interaction Constraints (ICs) are used to enforce a set of constraints only under certain conditions. Whenever the condition of an IC applies to a context (CDB) one of its resolvers must be applied to satisfy the IC. Take the following as an example:

```
ic HatesRobots(H,L,R,A,I):
  condition:
    statements:  ( C, activity(H,L,A) )
                 ( I, at(R), L )
    constraints: hasProperty(H,hatesRobots);
                 _possibleIntersection(C,I);
  resolver: C Before I [1,inf];
  resolver: C After  I [1,inf];
  resolver: socialCost := socialCost + 5.0;
```

This constraint only applies to inhabitants with property *hatesRobots* and when intervals *C* and *I* can possibly intersect. The function we use to calculate the intersection is common in the scheduling literature (Cesta, Oddi, and Smith, 2002) and assumes earliest start and end time for the intervals. If this condition holds one of the resolvers must be applied. The first two resolvers separate *C* and *I* in time. The third resolver allows to satisfy the constraint by paying a social cost. Social costs can either be used to determine the quality of a plan or to prune unacceptable plans by setting a maximum social cost (as was done in the initial context example).

We can express the meaning of an IC in logical notation

$$Condition \Rightarrow Resolver_1 \vee \ldots \vee Resolver_n$$

where $Condition$ is a CDB and $Resolver_i$ are sets of constraints. For convenience we use $Condition(c)$ and $Resolvers(c)$ to access the respective data from any IC c. Whenever a condition applies at least one resolver has to apply in order to satisfy the constraint:

$$Satisfied_{IC}(c, \Phi) \equiv \forall \sigma \in \theta(Condition(c), \Phi) : \\ Supports(\Phi, \sigma(Condition(c))) \\ \Rightarrow \bigvee_{r \in Resolvers(c)} Supports(\Phi, \sigma(r)).$$

Actions are also based on statements and constraints to model their conditions and effects. Consider these examples:

```
operator Move( R, L1, L2 ):
  preconditions: ( P , at(R) , L1 )
  effects:       ( E , at(R) , L2 )
  constraints:
    THIS Duration [T,inf];
```

```
    P Meets THIS; THIS Meets E;
    E Duration [1,inf];
    adjacent(L1,L2);
    movingTime(R,L1,L2,T);
```

Here, P and E are intervals that have an attached state-variable $at(R)$ that assigns the robot's position. Temporal constraints are used to determine the time it takes to move and when the assignment of $at(R)$ changes. Logical constraints are used to make sure both locations are adjacent and to decide the minimum time T it takes to move. Given the above initial context, the two robots could move to any location adjacent to the robot room. The second operator is used to perform tasks:

```
operator PerformTask( R, Task, Class, L ):
  preconditions:        ( P1, at(R), L )
    ( P2, state(Task,Class,L), waiting )
  effects:
    ( E1, state(Task,Class,L), attended )
    ( E2, state(Task,Class,L), finished )
    ( E3, busy(R), 1 )
  constraints: THIS Duration [T,inf];
      THIS During P1 [1,inf] [1,inf];
      THIS Equals E1; E1 Equals E3;
      P2 Meets E1;    E1 Meets E2;
      capability(R,Class);
      executionTime(R,Class,T);
```

It requires the robot to be at the right location and changes the state of the task from *waiting* to *attended* and finally to *finished*. Logical constraints use background knowledge to make sure the robot is capable of performing the task and to lookup the minimum time T it takes R to perform a task of this class. While the task is performed we assign a value of 1 to variable *busy(R)*, which is a reusable resource with a capacity of 1 to make sure that any robot only performs one task at a time.

More formally, an action $a = (\mathcal{P}_a, \mathcal{E}_a, \mathcal{C}_a)$ has a set of precondition statements $\mathcal{P}_a$, effect statements $\mathcal{E}_a$ and constraints $\mathcal{C}_a$ and can be applied to a CDB $\Phi = (\mathcal{F}_\Phi, \mathcal{C}_\Phi)$ iff $\mathcal{P}_a \subseteq \mathcal{F}_\Phi \wedge \mathit{Consistent}(\gamma(a, \Phi))$ where γ is the transition function (Ghallab, Nau, and Traverso, 2004) formulated in terms of CDBs: $\gamma(a, \Phi) = (\mathcal{F}_\Phi \cup \mathcal{E}_a, \mathcal{C}_\Phi \cup \mathcal{C}_a)$. A plan π is a sequence of actions and is applicable to a CDB Φ iff $(\forall a_i \in \pi : \mathcal{P}_{a_i} \subseteq \mathcal{F}_\Phi \cup \bigcup_{j \neq i} \mathcal{E}_{a_j}) \wedge \mathit{Consistent}(\gamma(\Phi, \pi))$, where $\gamma(\pi, \Phi) = (\mathcal{F}_\Phi \cup \bigcup_{a \in \pi} \mathcal{E}_a, \mathcal{C}_\Phi \cup \bigcup_{a \in \pi} \mathcal{C}_a)$. A planning problem $\Pi = (I, G, \mathcal{A})$ consists of an initial CDB I, a goal CDB G and a set of actions $\mathcal{A}$. A plan π is a solution to Π iff $\forall a \in \pi : a \in \mathcal{A}$ and π is applicable to I and $\mathit{Supports}(\gamma(I, \pi), G)$.

In classical planning, axioms provide a compact way to imply facts from states. ICs perform a similar function, as they imply constraints. Thibaux, Hoffmann, and Nebel (2003) show how compiling axioms into actions causes significantly larger domain formulations and plans, thus adding complexity to the planning problem. Compiling ICs into actions poses a similar issue. Let us assume, for instance, that we wanted to compile the IC *HatesRobots* into the *Move* action. This entails that every activity *(a, activity(H,L,A))* that can be matched to the condition has to be added to the preconditions. In addition we have to add a copy of the action for every possible combination of resolvers. Given just 10 matching activities in the initial context this would lead to 10^3 versions of *Move*. None of these can be told apart by a causal heuristic, since they all achieve the same effect. This forces the planner to pick a combination of resolvers together with an operator, which dilutes search and is an unnecessary over-commitment. In addition, the condition and resolvers of ICs may contradict each other, thus making it impossible to model both condition and resolver in an action. This is the case in the previous example, where *possibleIntersection(C,R)* contradicts the first two resolvers. As opposed to axioms, however, ICs imply constraints, not facts. Therefore, they cannot lead to sub-goals or threats to already achieved goals. The only way in which uncontrollable activities can affect the search space of causal reasoning, is if human activities appear in preconditions or effects of actions. Here we assume this not to be the case and can thus disregard uncontrollable activities and ICs during causal reasoning.

Planner

Algorithm 1 shows a basic constraint-based planning algorithm PLAN. It first uses CAUSALREASONING (line 3) to create a plan using heuristic forward planning with a combination of the Causal Graph (Helmert, 2006) and Fast Forward (Hoffmann and Nebel, 2001) heuristics that uses alternating search queues (as suggested by Helmert (2006)). Temporal propagation on goals is used to determine an order and extract a structure of requirements between goals. This enforces that goals that have an earlier start time have to be achieved first. Inside CAUSALREASONING all constraints and intervals are ignored which allows to solve this subproblem as a state-variable planning problem (Bäckström and Nebel, 1993) with some extensions, as e.g., requirements between goals (detailed by Köckemann, Pecora, and Karlsson (2014)). The plan is applied to the initial context (line 5) and RESOLVE tests the resulting CDB Φ for consistency wrt. to costs, temporal and resource constraints. ICs are resolved by RESOLVEICS (line 6) which is detailed in Algorithm 2 (explained in next section). If no conflict is found the plan π is returned as a solution, otherwise CAUSALREASONING is called again, resuming its search. If CAUSALREASONING fails to suggest a plan PLAN fails (line 4).

RESOLVECOSTS implements $\mathit{Satisfied}_C$ by simply calculating the involved costs and applying inequalities. RESOLVET&R implements $\mathit{Satisfied}_R$ by using a precedence constraint posting algorithm (Cesta, Oddi, and Smith, 2002) that includes $\mathit{Satisfied}_T$. RESOLVELOGICAL implements $\mathit{Satisfied}_L$ by using Prolog (Bratko, 2000) (given background knowledge that is part of domain and/or problem description). If an inconsistency is discovered at any point PRUNING is used (line 8) to determine the shortest sequence of actions that causes a conflict and to prune the search space of CAUSALREASONING accordingly. PRUNING uses binary search on π with RESOLVE and RESOLVEICS to find the shortest inconsistent plan.

Algorithm 1 already employs problem decomposition by separating causal reasoning from reasoning on constraints.

Our experiments will show how we can exploit this idea even further. The Algorithm is correct for problems that include costs, logical, temporal, resource and interaction constraints, since all sub-procedures that add constraints (i.e., RESOLVET&R and RESOLVEICS) test consistency of all constraint types that can be added.

The completeness of Algorithm 1 depends on CAUSALREASONING and PRUNING. The approach outlined above can be incomplete for certain domains, due to limitations of heuristic forward planning. One such limitation is the sequential approach to decision making that can lead to cases in which two actions that are applicable in combination cannot be applied in isolation. This could be remedied with a partial-order based approach (McAllester and Rosenblitt, 1991). Despite this, we justify our choice of heuristic forward planning with its performance in plan generation.

Algorithm 1 Constraint-based planning

Require: I, G - initial and goal CDB, $\mathcal{A}$ - set of actions
1: **function** PLAN($I, G, \mathcal{A}$)
2: **loop**
3: $\pi \leftarrow$ CAUSALREASONING($I, G, \mathcal{A}$)
4: **if** $\pi = Failure$ **then return** $Failure$
5: $\Phi \leftarrow \gamma(I, \pi)$
6: **if** RESOLVE(Φ) $\wedge$ RESOLVEICS(Φ) **then**
7: **return** π ▷ Success
8: **else** PRUNING(π, I)
9: **function** RESOLVE(Φ)
10: **if** $\neg$ (RESOLVECOSTS(Φ) $\wedge$ RESOLVELOGICAL(Φ) $\wedge$ RESOLVET&R(Φ)) **then return** $False$
11: **return** $True$

Resolving Interaction Constraints

Algorithm 2 shows how ICs are resolved. It first replaces every IC with a set of candidates matching the context (line 2-5). Then it calls a recursive function (line 6) that takes the first IC whose condition is satisfied (line 8-9), removes it, finds a working resolver and calls itself recursively (line 10-14). If the recursive call is successful or no IC applies we return $True$. If the recursive call returns $False$ we try the next resolver if one is available or return $False$ otherwise. The type of search that Algorithm 2 realizes is a form of a meta-CSP, where applicable ICs are variables and resolvers are possible values.

The described algorithm naturally handles cases where the executed human activity is only partially specified, since line 4 will match all applicable constraints in case the human activity is a variable. Consider, e.g., the initial context from earlier, where *(a2, activity(mother,study,A))* will be matched to every IC that requires $A = reading$ or $A = working$ or does not constrain A (as e.g., *HatesRobots*). Thus, the resulting plan will be consistent with all activities that could possibly occur and no plan will be generated if that is not possible. The computational price is a larger number of potentially applicable constraints n, which leads to more consistency tests in line 9 and a larger recursion depth. This may become expensive, since if m is the maximum number of resolvers of all ICs, Algorithm 2 has to consider n^m combinations of resolvers in the worst case. In practice, however, temporal and logical constraints in the condition of the ICs can keep this number relatively low, even for large problems.

Algorithm 2 Resolving interaction constraints

Require: Φ - a CDB
1: **function** RESOLVE-ICS(Φ)
2: **for** $c \in \mathcal{C}_\Phi$ where $Type(c) = IC$ **do**
3: Remove c from $\mathcal{C}_\Phi$
4: **for** $\sigma \in \theta(\mathcal{F}_{Condition(c)}, \mathcal{F}_\Phi)$ **do**
5: $\mathcal{C}_\Phi \leftarrow \mathcal{C}_\Phi \cup \{\sigma(c)\}$
6: **return** RESOLVESINGLEIC(Φ)
7: **function** RESOLVESINGLEIC(Φ)
8: **for** $c \in \mathcal{C}_\Phi$ where $Type(c) = IC$ **do**
9: **if** RESOLVE($\Phi \cup Condition(c)$) **then**
10: Remove c from $\mathcal{C}_\Phi$
11: **for** $Resolver \in Resolvers(c)$ **do**
12: $\Phi' \leftarrow (\mathcal{F}_\Phi, \mathcal{C}_\Phi \cup Resolver)$
13: **if** RESOLVE(Φ') **then**
14: **if** RESOLVESINGLEIC(Φ') **then**
15: **return** $True$
16: **return** $False$ ▷ No working resolver
17: **return** $True$ ▷ No constraint applies

Experimental Setup

The purpose of the experiments is to show that the proposed planner is capable of creating plans that satisfy a set of ICs for problem instances of a reasonable size. In addition we will show the potential benefits of problem decomposition and constraint-based planning by comparing to a slight modification of Algorithm 1.

We randomly generated complete human schedules in the following way: Starting at 6:00am, every activity lasts between 30 and 60 minutes. After filling a 16 hour time frame in this manner, the activity of every family member is set to *sleep* for the rest of the day. We used three different sets of humans $h1$, $h2$ and $h3$ containing 5, 7 and 12 inhabitants. The number of goals varies between 10 and 30 ($g10$ and $g30$) and goals are randomly distributed into the three goal batches (keeping the number of goals in each batch approx. even). We used two configurations of uncertainty $u0$ (no uncertainty) and $u1$ (with uncertainty). For each activity created with $u1$ there is a 10% chance that this activity will be only partially specified and we randomly pick between two and four random possible values.

To further exploit problem decomposition we observe that satisfiability of logical constraints cannot change by applying any action or IC, since they are evaluated wrt. static background knowledge that is provided with domain and problem description. Once they are propagated they cannot be violated anymore. In practice, this propagation will replace all actions and ICs with a set of substitutes that has been proven to satisfy logical constraints. This allows us to create an alternative to Algorithm 1, where we move RESOLVELOGICAL outside of the loop of PLAN. We refer to Algorithm 1 as $p1$ and to this alternative as $p2$. Finally, we

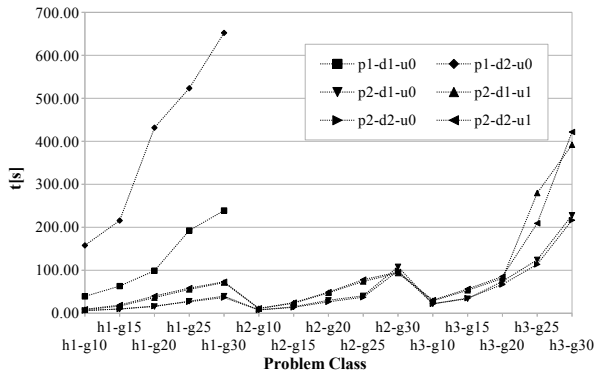

Figure 1: Average times on 20 random problems for 15 different configurations.

Problem	$p1$-$d1$	$p1$-$d2$	$p2$-$d1$	$p2$-$d2$
$h1$-$g10$	4.6 / 460.2	4.9 / 1118.3	4.6 / 140.7	4.9 / 155.4
$h1$-$g15$	6.6 / 731.8	6.9 / 1520.5	6.6 / 236.4	6.9 / 254.5
$h1$-$g20$	7.5 / 1008.2	8.2 / 2544.3	7.8 / 325.0	8.5 / 378.0
$h1$-$g25$	9.3 / 1677.3	9.8 / 3285.6	9.4 / 541.5	9.8 / 589.0
$h1$-$g30$	9.8 / 1861.4	10.6 / 3747.6	9.9 / 625.7	10.7 / 705.2

Table 1: Average number of applicable/tested IC conditions for $u0$ with $p1$ and $p2$ for domains $d1$ and $d2$.

created two domains $d1$ and $d2$ containing the first 6 and 9 ICs from the list presented earlier to evaluate how the number of ICs changes the problem solving time. For each configuration we created 20 random problems. We set a time limit of 30 minutes to solve each problem. Experiments were run on an Intel® Core™ i7-2620M CPU @ 2.70GHz x 4, 4GB RAM platform with a 3GB memory limit.

Experimental Results

Figure 1 shows the average problem solving times on 20 random problems for each class. Planner $p1$ shows poor performance, but using planner $p2$ we were able to solve problems in a reasonable time, even for the most difficult set that we used ($g30$, $h3$, $u1$). Comparing $u0$ to $u1$ we can see that the cost of uncertainty increases for the more difficult problems, which is not surprising, since more activities and goals are likely to increase the number of applicable ICs. An exception is $h2 - g25$ which takes less time with $u1$ than with $u0$. $u1$ has more potentially applicable ICs but slightly less actually applicable ones. This indicates possible gains from favorable variable orderings. In a similar way $h3 - g25$ has lower times for $d2$ than for $d1$. Our logs show that temporal reasoning and resource scheduling account for the difference which may be explained by ICs in the full set leading to fewer resource conflicts, which raises questions regarding the interplay between different constraint types.

Problem decomposition is enabled by the fact that we model each aspect of a domain with the appropriate type of constraint. In doing so we can separate reasoning about time/resources, logical constraints, causal dependencies among actions, ICs and costs from one another. This allows us to achieve planner $p2$, which significantly reduces the number of IC conditions that need to be tested (Algorithm 2 line 4). This effect is shown in Table 1, where we compare the number of applicable and tested IC conditions for a subset of problems.

There is another instance of problem decomposition in Algorithm 1, which lies in the fact that CAUSALREASONING proposes a complete plan π and only considers other constraints after this sub-problem has been solved. Running RESOLVE and RESOLVEICS for every decision considered by CAUSALREASONING would be a clear overhead, since even in cases where we find a conflict, PRUNING needs a logarithmic (since it uses binary search) number of tests (rather than linear) to find the action that caused the conflict.

Temporal expressiveness also contributes to the presented results. Removing the *possible intersection* constraint from the condition of ICs adds a large amount of combinations to the set of applicable ICs that need to be resolved even if they implicitly are resolved already. For $p2$ in Table 1 this would force Algorithm 2 to resolve all potentially applicable ICs, since they become applicable.

Except for one instance, all problems in our experiments were solvable within the allowed time of 30 minutes. In this one case, however, the limit was reached in PRUNING, which is explained by the fact that RESOLVEICS has to explore an exponential number of resolver combinations in the worst case. This could be counteracted (to a certain degree) by employing ordering heuristics for the choice of IC (line 2) and resolver (lines 4–5) in Algorithm 2. These correspond to variable and value ordering heuristics in a meta-CSP search.

Conclusions & Future Work

We have addressed the problem of considering partially specified schedules of uncontrollable agents in task-planning. We employ interaction constraints to model preferences of these agents in relation to plans. We propose a constraint-based approach that allows to decompose reasoning about ICs from plan generation. We demonstrate feasibility of the approach and the benefits of problem decomposition in a series of experiments where we planned a day for a set of robots in a house that is inhabited by a large family.

Our approach is particularly useful for human-aware planning. There are, however, still several open problems in this field: While we use cost constraints to impose maximum social costs, we have not yet considered optimization. Human schedules may not be known (at all) at planning time. This would require us to resolve ICs during execution, which poses additional challenges (Pecora et al., 2012; Myers, 1999; Lemai and Ingrand, 2004), such as re-planning. Another interesting direction is to extend of ICs to allow new goals as resolvers. If, for instance, a robot happens to be in the same room as someone who is working, we may want it to assist, even if this was not one of the original goals. Combined with the previous on-line extension this would allow very interesting applications.

Acknowledgement. This work was supported by the Swedish Research Council project "Human-aware task planning for mobile robots".

References

Alami, R.; Chatila, R.; Clodic, A.; Fleury, S.; Herrb, M.; Montreuil, V.; and Sisbot, E. A. 2006. Towards Human-Aware cognitive robots. In *Proc of the 5th Int Cognitive Robotics Workshop (COGROB 2006)*.

Allen, J. 1984. Towards a general theory of action and time. *Artificial Intelligence* 23:123–154.

Bäckström, C., and Nebel, B. 1993. Complexity Results for SAS+ Planning. *Computational Intelligence* 11:625–655.

Beaudry, E.; Kabanza, F.; and Michaud, F. 2010. Planning with concurrency under resources and time uncertainty. In *Proceedings of the 19th European Conf on Artificial Intelligence (ECAI 2010)*.

Benton, J.; Coles, A. J.; and Coles, A. 2012. Temporal planning with preferences and time-dependent continuous costs. In *Proceedings of the 22nd International Conference on Automated Planning and Scheduling (ICAPS 2012)*.

Blythe, J. 1994. Planning with external events. In *Proceedings of the 10th Conference on Uncertainty in Artificial Intelligence*.

Bratko, I. 2000. *Prolog Programming for Artificial Intelligence*. Addison Wesley.

Bresina, J.; Jónsson, A.; Morris, P.; and Rajan, K. 2005. Activity Planning for the Mars Exploration Rovers. In *Proceedings of the 15th International Conference on Automated Planning and Scheduling (ICAPS 2005)*.

Cesta, A.; Oddi, A.; and Smith, S. 2002. A Constraint-Based Method for Project Scheduling with Time Windows. *Journal of Heuristics* 8:109–136.

Chien, S.; Knight, R.; Stechert, A.; Sherwood, R.; and Rabideau, G. 2000. Using iterative repair to improve responsiveness of planning and scheduling. In *Proceedings of the 5th International Conference on Artificial Intelligence Planning and Scheduling (AIPS 2000)*.

Cirillo, M.; Karlsson, L.; and Saffiotti, A. 2009. A human-aware robot task planner. In *Proceedings of the 19th International Conference on Automated Planning and Scheduling (ICAPS 2009)*.

Coles, A.; Fox, M.; Long, D.; and Smith, A. 2008. Planning with problems requiring temporal coordination. In *Proceedings of the 23rd national conference on Artificial intelligence (AAAI 2008)*.

Do, M. B., and Kambhampati, S. 2003. SAPA: A multi-objective metric temporal planner. *Journal of Artificial Intelligence Research* 20:155–194.

Fratini, S.; Pecora, F.; and Cesta, A. 2008. Unifying Planning and Scheduling as Timelines in a Component-Based Perspective. *Archives of Control Sciences* 18(2):231–271.

Freksa, C. 1992. Temporal Reasoning Based on Semi-Intervals. *Artificial Intelligence* 54:199–227.

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

Helmert, M. 2006. The fast downward planning system. *Journal of Artificial Intelligence Research* 26(1):191–246.

Hoffmann, J., and Nebel, B. 2001. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research* 14:2001.

Karlsson, L. 2001. Conditional progressive planning under uncertainty. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI 2001)*.

Köckemann, U.; Pecora, F.; and Karlsson, L. 2014. Expressive planning through constraints. In *Proceedings of the 12th Scandinavian Conference on Artificial Intelligence (SCAI 2014)*.

Lemai, S., and Ingrand, F. 2004. Interleaving temporal planning and execution in robotics domains. In *Proceedings of the 19th National Conference on Artifical Intelligence (AAAI 2004)*.

Mausam, M., and Weld, D. S. 2008. Planning with durative actions in stochastic domains. *Journal of Artificial Intelligence Research* 31:33–82.

McAllester, D., and Rosenblitt, D. 1991. Systematic nonlinear planning. In *In Proceedings of the 9th National Conference on Artificial Intelligence (AAAI 1991)*.

Meiri, I. 1996. Combining Qualitative and Quantitative Constraints in Temporal Reasoning. In *Artificial Intelligence*, 260–267.

Montreuil, V.; Clodic, A.; Ransan, M.; and Alami, R. 2007. Planning human centered robot activities. In *IEEE International Conference on Systems, Man and Cybernetics (SMC 2007)*.

Myers, K. L. 1999. CPEF: A Continuous Planning and Execution Framework. *AI Magazine* 20(4):63–69.

Pecora, F.; Cirillo, M.; Dell'Osa, F.; Ullberg, J.; and Saffiotti, A. 2012. A Constraint-Based Approach for Proactive, Context-Aware Human Support. *Journal of Ambient Intelligence and Smart Environments* 4(4):347–367.

Thibaux, S.; Hoffmann, J.; and Nebel, B. 2003. In defense of PDDL axioms. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI 2003)*.

Tsamardinos, I.; Vidal, T.; and Pollack, M. E. 2003. CTP: A New Constraint-Based Formalism for Conditional, Temporal Planning. *Constraints* 8(4):365–388.