Causality-Based Reasoning for Cognitive Factories*

Esra Erdem and **Kadir Haspalamutgil** and **Volkan Patoglu**
Faculty of Engineering and Natural Sciences, Sabancı University, İstanbul, Turkey

Tansel Uras†
Department of Computer Science, University of Southern California, Los Angeles, USA

Abstract

We propose the use of causality-based formal representation and automated reasoning methods to endow multiple teams of robots in a factory, with high-level cognitive capabilities, such as, optimal planning and diagnostic reasoning. We introduce algorithms for finding optimal decoupled plans and diagnosing the cause of a failure/discrepancy (e.g., robots may get broken or tasks may get reassigned to teams). We discuss how these algorithms can be embedded in an execution and monitoring framework, and show their applicability on an intelligent painting factory scenario.

Introduction

As conventional manufacturing and assembly systems fall short of responding to ever increasing market demands for customized and variant-rich products in a cost effective manner, new approaches for automated fabrication of customized product become well-motivated. The cognitive factory concept (Zaeh et al. 2009; Beetz, Buss, and Wollherr 2007) is such a paradigm shift that promises significant advantages over conventional manufacturing by balancing the flexibility and efficiency demands in automation, while simultaneously achieving a high-degree of reliability. In particular, cognitive factories aim to endow manufacturing system with high-level reasoning capabilities in the style of cognitive robotics, such that these systems become capable of planning their own actions, reconfiguring themselves to allow fabrication of a wide range of parts and to react to change in demands, detecting failures during execution, diagnosing the cause of these failures and recovering from such failures. Since cognitive factories can plan their own actions and self-reconfigure, they can rapidly respond to changing customer needs and customization requests, demonstrating the necessary flexibility while maintaining cost-effectiveness compared to human workshops. Moreover, thanks to fault-awareness, diagnostic reasoning and failure recovery features of cognitive factories, these systems enable a high-degree of reliability comparable to those of mass production systems.

We present a generic and modular planning and execution framework for a cognitive factory that involves complex tasks performed concurrently/cooperatively by multiple teams of robots by efficiently using the shared resources. We focus on two crucial aspects of a cognitive factory setting: planning and decision-making for reconfigurable networked robots for efficient use of shared resources (e.g., workforce, time, product components) and diagnostic reasoning and monitoring for fault-tolerance (e.g., preventing action failures and recovering from failures when they occur). We propose the use of causality-based formal representation (e.g., action language $\mathcal{C}+$ (Giunchiglia et al. 2004)) and automated reasoning methods and tools (e.g., the causal reasoner CCALC (McCain and Turner 1997)) to endow multiple teams of robots in a factory, with such high-level cognitive capabilities. In particular, we introduce novel algorithms for finding optimal decoupled plans and for diagnosing the cause of a failure/discrepancy (e.g., robots may get broken or tasks may get reassigned to teams). To be able to coordinate networked robots and diagnose and handle failures in a dynamic setting, we embed these algorithms modularly in a generic execution and monitoring framework that allows reusability of computed plans in case of failures. The proposed generic framework is applied, in particular, to cognitive factory scenarios.

Our cognitive factory framework possesses core characteristics of a reconfigurable manufacturing system. In particular, the approach not only increases the speed of responsiveness of manufacturing systems to unpredicted events, such as sudden market demand changes or unexpected machine failure, but also facilitates a quick production launch of new products and flexible customization of existing products in product family. Reasoning about its resources, the proposed approach adjusts itself to provide exactly the functionality and production capacity needed, maximizing the system productivity with the available resources. In that sense, our work plays an important role towards Cognitive Technical Systems (CTS)—systems "that know what they are doing" (Brachman 2002).

This paper summarizes our work on cognitive factories (Erdem et al. 2012).

Related Work

There are contributions in the literature that are relevant to some components of our framework. For instance, in (Ertelt et al. 2009) a method is presented that integrates a global planning system with the domain specific machining planning system of (Shea et al. 2010) and an ad-hoc perception mechanism. In particular, the manufacturing domain is represented in PDDL and a classical planner is utilized to find a sequence of actions to reach the goal. The main role of

*This work is partially supported by TUBITAK Grant 111E116.

†The work carried out while studying at Sabancı University.

global planner is to augment the local machining plans with transportation and handling operations. The perception system is used for inspection of machined parts and triggers re-planning of the global planner when faulty parts are detected. Unlike our framework, this approach does not consider optimal planning for multiple teams of robots, cooperation of robots/teams, efficient use of resources, or diagnostic reasoning.

There are also recent contributions in the literature that are relevant to cognitive factories (Shea 2010; Bannat et al. 2011) that complements our framework. The approaches for generative CNC machining planning using shape grammars and for automated fixture design to enable autonomous fabrication of customized part geometries (Shea et al. 2010), methods that enable human-robot cooperation in a cognitive factory setting (Lenz et al. 2008), and the model-based approach that computes success probabilities of plans utilizing online observations (Maier et al. 2010) can be listed as representatives of these interesting contributions.

Finally, several alternative approaches have been proposed for modeling of cognitive manufacturing systems. In particular, in (Ruhr, Pangercic, and Beetz 2008) the use of structured reactive controllers and transformational modeling are advocated, while in (Rungger et al. 2008) hierarchical hybrid modeling and control are proposed as a viable solution. These approaches are not comparable to our system, since they attack different challenges and emphasize different aspects of modeling of cognitive factories.

A Cognitive Painting Factory Scenario

We consider a painting factory with multiple teams of robots, where each team is located in a separate workspace collectively working toward completion of an assigned task: painting, waxing and stamping a given number of boxes. Each workspace is depicted as a grid, contains an assembly line along the north wall to move the boxes and a pit stop area where the worker robots can change their end-effectors.

The teams are heterogenous. Each team is composed of two types robots with different capabilities: worker robots and a single charger robot. Worker robots operate on boxes, they can configure themselves for different stages of process, and they can be exchanged between teams; charger robots maintain the batteries of workers and monitor teams plan, and cannot be exchanged between teams. The robots can move horizontally or vertically.

The teams act as autonomous cognitive agents; therefore, each team makes its own plan to complete its own designated task. On the other hand, to make more efficient use of shared resources (e.g., robots), teams can exchange robots: at any step, a team can lend one of its worker robots through their pit stop such that after a transportation delay the worker robot shows up in the pit stop of a borrowing team. Therefore, given the initial state of each workspace and the designated tasks for each team (e.g., how many boxes of which colors to paint), the goal is for all the teams to complete these tasks in a minimum number of steps. We propose to reach this goal by means of an optimal decoupled planning algorithm.

Once an optimal decoupled plan is computed, the teams start executing it. However, during a plan execution, it is possible that a robot gets broken so that it can not dock to another robot, or that tasks assigned to teams (e.g., the number of orders) are modified. In such cases, the goal is to diagnose the cause of the failure or discrepancy, and find an optimal (decoupled) plan for recovery. We propose to reach this goal by means of a diagnosis algorithm.

Both the optimal planning algorithm and the diagnosis algorithm we propose are based on a causality-based approach. In this approach, for each team, we represent the actions of its robots and the other changes in its workspace in the high-level causality-based action language $\mathcal{C}+$ (Giunchiglia et al. 2004). With this high-level formulation, each team can find answers to queries about the existence of plans or the causes of failures, using the causal reasoner CCALC (McCain and Turner 1997). By utilizing such a causality-based approach for reasoning about plans for each team and reasoning about failures in each workspace/team, we introduce novel algorithms to compute an optimal decoupled plan of actions for all the teams, and to compute a minimal diagnosis for a failure. Moreover, in case of failures/discrepancies, we show how these two algorithms can be embedded in an execution and monitoring framework to be able to respond robustly to failures/discrepancies. Note that reasoning is performed in each team/workspace independent from the others in a decentralized fashion. Thanks to this decentralized approach, our algorithms are scalable to domains with large number of workspaces.

The algorithms introduced in this paper are not specific to the painting factory scenario described above. They are applicable to various cognitive factory scenarios that involve multiple teams of robots and single robot exchange between these teams. The underlying reasoning methodology of our approach, causality-based reasoning, is also generic and has been successfully applied to cognitive robotics including demonstrations on physical robots (Erdem et al. 2011; Aker et al. 2011; Caldiran et al. 2009; Erdem and Patoglu 2012; Havur et al. 2013).

Representation of a Cognitive Factory

We represent a cognitive factory in the high-level action language $\mathcal{C}+$ (Giunchiglia et al. 2004), which is a logic-based formalism based on causality. Using this language, we can describe preconditions and (conditional) effects of actions, as well as indirect effects of actions and static/dynamic constraints. We can describe true concurrency (where actions cannot be serialized) and nondeterministic effects of actions (where we are not sure about the outcome of an action).

For instance, in the painting factory described above, the action of a worker robot `W` working on a box `B` is formalized in $\mathcal{C}+$ by "causal laws" that describe direct effects of this action (e.g., incrementing the work stage `WS` of a box `B`):

```
workOn(W,B) causes workDone(B)=WS
  if workDone(B)=WS-1.
```

and its preconditions (e.g., a worker robot `W` cannot work on a box `B` that still has wet paint):

```
nonexecutable workOn(W,B) if wetpaint(B).
```

We can also describe change that does not directly involve an action of a robot. For instance, we formalize that a box with wet paint gets dry, by the causal law

```
caused -wetpaint(B) after wetpaint(B).
```

Similarly, we can express that a worker robot can work on a box if it is right next to the assembly line and it is aligned with the box, that a worker robot swaps its end-effector only if it is in the pit area, and other preconditions of the worker's actions. We can also formalize the preconditions and effects of a charger's actions.

Concurrent actions are allowed unless specified otherwise. For instance, we can express that a charger robot cannot dock to a worker robot for charging while moving to some location:

```
nonexecutable dock(C,W) & move(C,D).
```

Similarly, we prevent some other concurrent actions, e.g., a worker cannot work on a box while the assembly line is shifting.

Causality-Based Reasoning for Finding Optimal Decoupled Plans

Once an action domain is described by a set of causal laws as shown above, we can present it to CCALC and ask CCALC "queries" about the existence of plans or the causes of observed failures. CCALC transforms the given domain description and the query into a set of propositional formulas, calls a SAT solver (e.g., MANYSAT (Hamadi, Jabbour, and Sais 2009)) to find a model of these formulas, and extracts an answer to the given query from this model. For more detailed information about CCALC, we refer the reader to (McCain and Turner 1997; Giunchiglia et al. 2004).

For instance, the following query asks for a plan whose length is at most 100, for a team with one worker (`w1`) and one charger (`c1`):

```
:- query
maxstep :: 100;
0: % INITIAL STATE
 % no charger robot is docked
 %    to a worker robot
 [/\C /\W |-docked(C,W)],
 % no block has wetpaint
 [/\B | -wetpaint(B)=0],
 % worker is at (1,3)
 xpos(w1)=1, ypos(w1)=4,
 % charger is at (1,1)
 xpos(c1)=1, ypos(c1)=1,
 % boxes are not yet processed
 [/\B | linePos(B)=B+lineLength],
 [/\B | workDone(B)=0];
maxstep: %GOAL
 % boxes are painted
 linePos(maxBox)=0,
 [/\B | workDone(B)=3].
```

If we replace `maxstep :: 100` with `maxstep :: 18..100`, then the query asks for a *shortest* plan whose length is at least 18 and at most 100. In that case, with the domain description whose some parts are briefly explained above in the previous section, CCALC finds a shortest plan of length 29 using the parallel SAT solver MANYSAT.

Suppose now that the goal is to complete all the assigned tasks of all the teams in a minimum number of steps, under the assumption that teams can exchange robots. For a plan length k, a team is a lender if it can complete its task on its own in k steps; a borrower if it can not complete its task on its own in k steps. Assuming that a team can not lend a robot and borrow a robot in a plan and that a team can not lend or borrow more than one robot, we designed an intelligent algorithm that finds an optimal decoupled plan by efficiently using the shared resources. This algorithm relies on the following two sorts of queries (asked to each team) to decouple plans and to orchestrate robot exchanges among the teams:

- Can the goal be achieved in k steps, while lending a robot before step k_0?
- Can the goal be achieved in k steps, while borrowing a robot after step k_0?

In decoupled planning, once answers to such queries are collected, the goal is to match each borrowing team with a different lending team so that the matched teams agree on lend/borrow times. Let lend_i denote the time step at which Team i can lend a robot (i.e., Team i answers the first query above affirmatively for $k = \mathit{lend}_i$). Let borrow_i denote the time step at which Team j can borrow a robot (i.e., Team j answers the second query above affirmatively for $k = \mathit{borrow}_i$). Let *delay* denote the transportation delay. Then, there is a matching between the lending team and the borrowing team, if $\mathit{lend}_i + \mathit{delay} \leq \mathit{borrow}_j$. Based on this observation, the optimal decoupled planning algorithm uses binary search for each team to find valid lend/borrow times.

Diagnostic Reasoning in a Cognitive Factory

In the painting factory domain, a robot may get broken and thus may not succeed docking to another robot or working on the boxes. We assume that a global sensor can detect if robots are docked/undocked (but cannot detect which robot is broken), and the work stage. Once such discrepancies ("exceptions") are noticed, the goal is to find their possible causes (i.e., which robots can be broken) so that an external agent (e.g., human or some other robot) can inspect the possibly broken robots and repair them. To take into account such discrepancies, we modify the domain description, and introduce an algorithm that computes *minimal diagnoses* by means of asking CCALC prediction queries with temporal constraints over the modified domain description.

In the domain description, we introduce a new fluent `broken(R)` to describe that a robot `R` may get broken at any step:

```
caused broken(R)
  if broken(R) after -broken(R).
```

and modify the causal laws describing the effects of relevant actions, e.g., a charger robot `C` docking to a worker robot `W`:

```
dock(C,W) causes docked(C,W)
  if -broken(C) & -broken(W).
```

Suppose that after a sequence $\langle A_0, ..., A_n \rangle$ of (concurrent) actions is executed at a state S, a discrepancy is observed be-

tween the expected state (with respect to the domain description) and the observed state S' (obtained by sensors). In the painting domain, the causes of such discrepancies are due to broken robots; therefore, we can identify possible diagnoses by sets of possibly broken robots. We can find the set C of all minimal sets of at most k broken robots by Algorithm 1. The outer while loop of Algorithm 1 considers cases where only the first $m = n, n-1, ..., 1$ actions are completely executed. Then, the inner while loop of Algorithm 1 checks (through a query to CCALC) for every subset r of at most $i = 1, 2, ..., k$ robots whether the execution of $A_0, ..., A_m$ of actions at S leads to S' where the robots in r are broken. If CCALC returns a positive answer to the query, then two important information becomes available: 1) an explanation as to when the robots in r may have got broken during the execution of these actions, 2) which actions in $A_{m+1}, ..., A_n$ are executed.

For instance, consider the execution of a single concurrent action `dock(c1,w3), workOn(w3,2)` at a state where `c1` is not docked to `w3` and Box `2` is waxed (i.e., stage is `2`). After this action is executed, we observe an unexpected state where `c1` is not docked to `w3` and Box `2` is stamped (i.e., stage is `3`). To find a minimal diagnosis, Algorithm 1 checks whether a single robot might be broken ($n = 1$, $k = 1$), by a CCALC query.

It is important to note here that, since broken robots are viewed and formulated in the domain description as "exceptions", specifying these exceptions in queries does not lead to inconsistencies, due to the nonmonotonic semantics of $\mathcal{C}+$.

Once broken robots are identified by Algorithm 1 as part of our execution and monitoring algorithm, an external agent can repair them. For that, we modify the domain description further by introducing a new action to repair broken robots. We also add causal laws to ensure that a robot does not perform any other actions while being repaired, or when it is broken. With the modified domain description, the execution and monitoring algorithm can ask CCALC to compute a plan to reach the goal from the observed state S' where discrepancy is detected. Here, we also specify the broken robots as part of S' so that they get repaired as part of replanning.

Algorithm 1 DIAGNOSE

Input: A state s, a sequence $A_0, ..., A_n$ of actions executed at s, an observed state s', a nonnegative integer k
Output: Current state for a team, potentially updated by robot breakdown information

```
R ← set of unbroken robots at state s;
C ← set of sets of at most k candidate robots to explain the anomalies, empty initially;
m := n; holds := false;
while m ≥ 0 and C = ∅ do
    // Find minimal sets of at most k possibly broken robots, assuming that actions A_0, ..., A_m are completely executed
    i := 1;
    while i ≤ k and C = ∅ do
        for all set r of i robots in R do
            s_r ← modify s' by making robots in r broken;
            holds ← check if executing A_0, ..., A_m (and possibly a subset of each action in A_{m+1}, ..., A_n) at s results in s_r;
            if holds then
                C := C ∪ {r};
        i++;
    m--;
x ← inspect the sets of robots in C to find the broken robots, and modify the state s' by specifying the broken robots;
return x;
```

Embedding Diagnostic Reasoning in an Execution Monitoring Framework

Let us demonstrate how our diagnosis algorithm and optimal decoupled planning algorithm can be embedded effectively in an execution and monitoring framework.

Algorithm 2 presents the overall execution and monitoring algorithm. First, for each team i, INIT tries to compute a plan $P_X[i]$ of length at most k_X; if such a plan is computed (i.e., the team may be able to spare a robot) then the team is designated as a lender; otherwise, it is designated as a borrower. After that, Algorithm 2 tries to find an overall plan with minimum number of steps, calling FIND_OPTIMAL_PLAN; if such a plan is found, then it is executed by the teams. During the execution, if a discrepancy is detected between the observed state of the world and the expected state, then Algorithm 2 tries to diagnose the cause of the discrepancy in terms of minimum number of broken robots, calling DIAGNOSE. If one of the robots is found as broken, then Algorithm 2 asks CCALC for a new plan that may include repair of the broken robot. Otherwise, the algorithm asks CCALC for a new plan from the observed state. Note that examples in which new orders arrive for teams are also handled via replanning in Algorithm 2.

To show the applicability of this framework to various kinds of robotic manipulators, we have experimented with some scenarios (Erdem et al. 2012). We have also implemented a cognitive factory scenario using dynamic simulation of Kuka youBots. A video of this implementation is available at http://cogrobo.sabanciuniv.edu/demos/cogfactory/youBot_planning.flv.

Conclusion

We have introduced two algorithms utilizing the formalisms, methods and tools of causal reasoning to endow multiple teams of self-reconfigurable robots with high-level reasoning capabilities. One of the algorithms computes optimal decoupled plans for multiple teams of robots that can exchange robots, and the other algorithm computes minimal diagnoses for failures in terms of number of broken robots.

We have shown the applicability and usefulness of these algorithms, embedded in a generic execution and monitoring framework, in a cognitive painting factory scenario, which provides a good case study towards future intelligent factories.

Algorithm 2 EXECUTE&MONITOR

Input: An action domain description $\mathcal{D}$, a nonnegative integer k, n planning problems $\mathcal{P}_1$, $\mathcal{P}_2, \ldots, \mathcal{P}_n$ (one for each team) with initial states $s_1, s_2, \ldots, s_n$ and goal states $g_1, g_2, \ldots, g_n$, and a transportation delay t_d

Output: Achieve the goals of all teams in minimum time steps

// Let X be a tuple consisting of a plan length k_X; and, for each team i, a plan $P_X[i]$ of length k_X, team role $role_X[i]$, and lower and upper bounds, $l_X[i]$ and $u_X[i]$, on the earliest/latest lend/borrow times.

$k_X := k$;
for all teams i **do**
  $P_X[i], role_X[i], l_X[i], u_X[i] \leftarrow$ INIT($\mathcal{D}$, k_X, $\mathcal{P}_X[i]$);
while $k_X > 0$ **do**
  $X \leftarrow$ FIND_OPTIMAL_PLAN($\mathcal{D}$, $\mathcal{P}_1$, $\mathcal{P}_2, \ldots, \mathcal{P}_n$, t_d, X);
  replan := *false*;
  while $\neg$*replan* $\wedge$ $k_X > 0$ **do**
    $k_X := k_X - 1$;
    for all teams i **do**
      $A_i, c_i, e_i, o_i \leftarrow$ extract from $P_X[i]$ the actions to be executed, the current state, the expected state after A_i and the observed state after A_i;
      $l_X[i], u_X[i], role_X[i] \leftarrow$ update the bounds and roles;
      updated := *false*;
      if $o_i \neq e_i$ **then**
        updated := *true*;
        $s_i \leftarrow$ DIAGNOSE($\mathcal{D}, \mathcal{P}_i, c_i, o_i$);
      if new order of boxes **then**
        updated := *true*;
        $\mathcal{P}_i \leftarrow$ modify the planning problem;
      if *updated* **then**
        replan := *true*;
        $s_i \leftarrow$ obtain the current state
        $P_X[i], role_X[i], l_X[i], u_X[i] \leftarrow$ INIT($\mathcal{D}, k_X, \mathcal{P}_i$);

References

Aker, E.; Erdogan, A.; Erdem, E.; and Patoglu, V. 2011. Causal reasoning for planning and coordination of multiple housekeeping robots. In *Proc. of LPNMR*.

Bannat, A.; Bautze, T.; Beetz, M.; Blume, J.; Diepold, K.; Ertelt, C.; Geiger, F.; Gmeiner, T.; Gyger, T.; Knoll, A.; Lau, C.; Lenz, C.; Ostgathe, M.; Reinhart, G.; Roesel, W.; Ruehr, T.; Schuboe, A.; Shea, K.; Stork genannt Wersborg, I.; Stork, S.; Tekouo, W.; Wallhoff, F.; Wiesbeck, M.; and Zaeh, M. 2011. Artificial cognition in production systems. *IEEE Transactions on Automation Science and Engineering* 8(1):148 –174.

Beetz, M.; Buss, M.; and Wollherr, D. 2007. CTS - What is the role of artificial intelligence? In *Proc. of KI*, 19–42.

Brachman, R. J. 2002. Systems that know what they're doing. *IEEE Intelligent Systems* 17(6):67–71.

Caldiran, O.; Haspalamutgil, K.; Ok, A.; Palaz, C.; Erdem, E.; and Patoglu, V. 2009. Bridging the gap between high-level reasoning and low-level control. In *Proc. LPNMR*, 342–354.

Erdem, E., and Patoglu, V. 2012. Applications of action languages in cognitive robotics. In *Correct Reasoning*, 229–246.

Erdem, E.; Haspalamutgil, K.; Palaz, C.; Patoglu, V.; and Uras, T. 2011. Combining high-level causal reasoning with low-level geometric reasoning and motion planning for robotic manipulation. In *Proc. of ICRA*.

Erdem, E.; Haspalamutgil, K.; Patoglu, V.; and Uras, T. 2012. Causality-based planning and diagnostic reasoning for cognitive factories. In *Proc. of ETFA*.

Ertelt, C.; Shea, K.; Pangercic, D.; Ruhr, T.; and Beetz, M. 2009. Integration of perception, global planning and local planning in the manufacturing domain. In *Proc. of ETFA*, 1 –8.

Giunchiglia, E.; Lee, J.; Lifschitz, V.; McCain, N.; and Turner, H. 2004. Nonmonotonic causal theories. *AIJ* 153:49–104.

Hamadi, Y.; Jabbour, S.; and Sais, L. 2009. Control-based clause sharing in parallel sat solving. In *Proc. of IJCAI*, 499–504.

Havur, G.; Haspalamutgil, K.; Palaz, C.; Erdem, E.; and Patoglu, V. 2013. A case study on the tower of hanoi challenge: Representation, reasoning and execution. In *Proc. of ICRA*.

Lenz, C.; Nair, S.; Rickert, M.; Knoll, A.; Rosel, W.; Gast, J.; Bannat, A.; and Wallhoff, F. 2008. Joint-action for humans and industrial robots for assembly tasks. In *Proc. of ROMAN*, 130–135.

Maier, P.; Sachenbacher, M.; Rhr, T.; and Kuhn, L. 2010. Automated plan assessment in cognitive manufacturing. *Advanced Engineering Informatics* 24(3):308 – 319. The Cognitive Factory.

McCain, N., and Turner, H. 1997. Causal theories of action and change. In *Proc. of AAAI/IAAI*, 460–465.

Ruhr, T.; Pangercic, D.; and Beetz, M. 2008. Structured reactive controllers and transformational planning for manufacturing. In *Proc. of ETFA*, 97–104.

Rungger, M.; Ding, H.; Paschedag, T.; and Stursberg, O. 2008. Hierarchical hybrid modeling and control of cognitive manufacturing systems. In *International Workshop on Cognition for Technical Systems*.

Shea, K.; Ertelt, C.; Gmeiner, T.; and Ameri, F. 2010. Design-to-fabrication automation for the cognitive machine shop. *Advanced Engineering Informatics* 24(3):251 – 268. The Cognitive Factory.

Shea, K., ed. 2010. *Special Issue in Cognitive Robotics*, volume 24. Advanced Engineering Informatics, Elsevier.

Zaeh, M.; Beetz, M.; Shea, K.; Reinhart, G.; Bender, K.; Lau, C.; Ostgathe, M.; Vogl, W.; Wiesbeck, M.; Engelhard, M.; Ertelt, C.; Rhr, T.; Friedrich, M.; and Herle, S. 2009. The cognitive factory. In *Changeable and Reconf. Manufacturing Systems*. 355–371.