AI Planning

4. Applications

A Few Problems We Can Solve (and Which Some People Care About)

Álvaro Torralba, Cosmina Croitoru

SAARLAND UNIVERSITY
COMPUTER SCIENCE

Winter Term 2018/2019

Thanks to Prof. Jörg Hoffmann for slide sources

Agenda

Motivation

. . . well, does anybody need to be motivated?

→ I'm presuming that "Applications" sounds better than "The expressive power of merge-and-shrink abstractions" . . .

Applications are important:

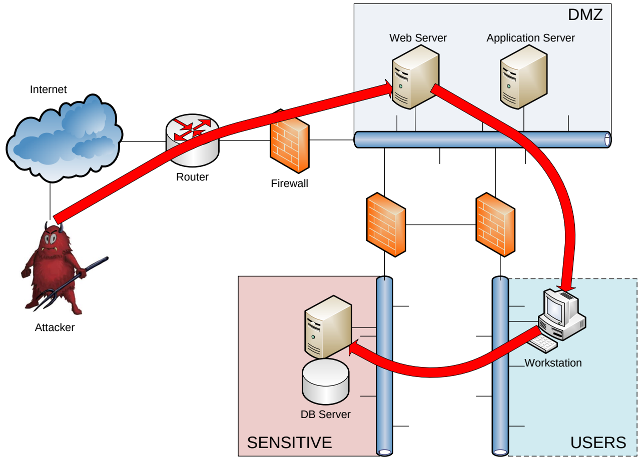
- Validate research ideas and techniques.
- Source of new research problems to consider.
- Source of useful benchmark examples to evaluate algorithms.

→ **FAI BSc/MSc/HiWi Jobs:** All three application areas here are major ongoing/future research efforts in FAI.

Our Agenda for This Chapter

2. **Simulated Penetration Testing:** Simulating hackers (well, simple versions thereof) for automated network security testing.
3. **Natural Language Generation:** Turning a language grammar and an intended meaning into a sentence.
4. **Modular Printing System Control:** How to control all printers that could possibly be built.

Network Hacking



Penetration Testing (Pentesting)

Pentesting

Actively verifying network defenses by conducting an intrusion in the same way an attacker would.

- Well-established industry (roots back to the 60s).
- Points out specific dangerous attacks (as opposed to vulnerability scanners).
- Pentesting tools sold by security companies, like Core Security.
  $\rightarrow$ Core IMPACT (since 2001); Immunity Canvas (since 2002); Metasploit (since 2003).
- Run security checks launching exploits.
- Core IMPACT uses FF (cf. **Chapter 9**) for automation since 2010.

Motivation for Automation

Motivation for Automation: Wrap-Up

Simulated penetration testing serves to:

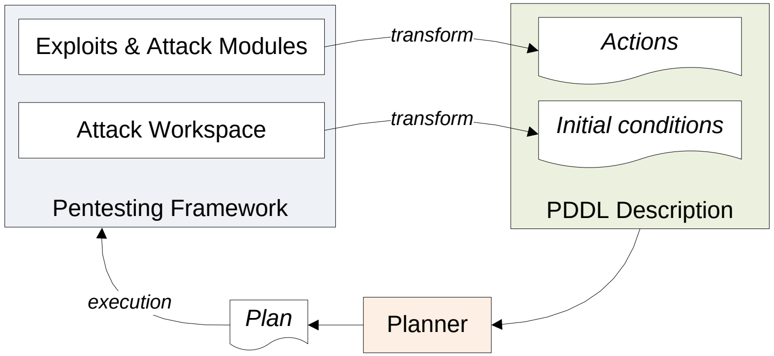
- Reduce human labor.
- Increase testing coverage:
  - Higher testing frequency.
  - Broader tests trying more possibilities.
- Deal with the dynamics of pentesting:
  - More exploits.
  - New tools used in attacks (Client-Side, WiFi, WebApps, ...).

$\rightarrow$ The aim is to automate pentesting, so that the attacks can continuously be run in the background, thus decreasing human labor while allowing broad coverage of complex attack possibilities.

The Turing Test, Revisited

Simulated Pentesting at Core Security

Core IMPACT system architecture:



→ In practice, the attack plans are being used to point out to the security team where to look.

Core Security PDDL

Object Types:

network	operating_system
host	OS_version
port	OS_edition
port_set	OS_build
application	OS_servicepack
agent	OS_distro
privileges	kernel_version

Core Security PDDL, ctd.

Predicates expressing connectivity:

```
(connected_to_network ?s - host ?n - network)
(IP_connectivity ?s - host ?t - host)
(TCP_connectivity ?s - host ?t - host ?p - port)
(TCP_listen_port ?h - host ?p - port)
(UDP_listen_port ?h - host ?p - port)
```

Core Security PDDL, ctd.

Predicates expressing configurations:

```
(has_OS ?h - host ?os - operating_system)
(has_OS_version ?h - host ?osv - OS_version)
(has_OS_edition ?h - host ?ose - OS_edition)
(has_OS_build ?h - host ?osb - OS_build)
(has_OS_servicepack ?h - host ?ossp - OS_servicepack)
(has_OS_distro ?h - host ?osd - OS_distro)
(has_kernel_version ?h - host ?kv - kernel_version)
(has_architecture ?h - host ?a - OS_architecture)
(has_application ?h - host ?p - application)
```


Core Security PDDL, ctd.

Actions modeling exploits:

```
(:action HP_OpenView_Remote_Buffer_Overflow_Exploit
:parameters (?s - host ?t - host)
:precondition (and (compromised ?s)
  (and (has_OS ?t Windows)
    (has_OS_edition ?t Professional)
    (has_OS_servicepack ?t Sp2)
    (has_OS_version ?t WinXp)
    (has_architecture ?t I386))
  (has_service ?t ovtrcd)
  (TCP_connectivity ?s ?t port5053)
)
:effect(and (installed_agent ?t high_privileges)
  (increase (time) 10)
))
```


Core Security PDDL, ctd.

Actions allowing to reap benefits of exploits:

```
(:action Mark_as_compromised
:parameters (?a - agent ?h - host)
:precondition (installed ?a ?h)
:effect (compromised ?h)
)

(:action IP_connect
:parameters (?s - host ?t - host)
:precondition (and (compromised ?s)
  (exists (?n - network)
    (and (connected_to_network ?s ?n)
      (connected_to_network ?t ?n))))
:effect (IP_connectivity ?s ?t)
)
```


Core Security PDDL, ctd.

An attack plan:

```
 0: Mark_as_compromised localagent localhost
 1: IP_connect localhost 10.0.1.1
 2: TCP_connect localhost 10.0.1.1 port80
 3: Phpmyadmin Server_databases Remote Code Execution
        localhost 10.0.1.1
 4: Mark_as_compromised 10.0.1.1 high_privileges
 ...
14: Mark_as_compromised 10.0.4.2 high_privileges
15: IP_connect 10.0.4.2 10.0.5.12
16: TCP_connect 10.0.4.2 10.0.5.12 port445
17: Novell Client NetIdentity Agent Buffer Overflow
        10.0.4.2 10.0.5.12
18: Mark_as_compromised 10.0.5.12 high_privileges
```

Simulated Pentesting@Core Security: Remarks

History:

- Planning domain "of this kind" (less IT-level, including also physical actions like talking to somebody) first proposed by [Boddy *et al.* (2005)]; used as benchmark in IPC'08 and IPC'11.
- Presented encoding proposed by [Lucangeli *et al.* (2010)].
- Used commercially by Core Security in Core INSIGHT since 2010.

Do Core Security's customers like this?

- I am told they do.
- In fact, they like it so much already that Core Security is very reluctant to invest money in making this better . . .

Questionnaire

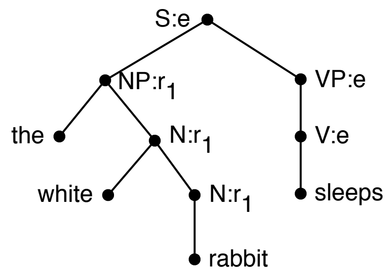
Question!

Is the current realization @Core Security really a simulation of what human hackers do?

(A): Yes. (B): No.

Research → FAI BSc/MSc/HiWi; Cooperation CISPA

Natural Language Generation (NLG)



- **Input:** Grammar, intended meaning.
- **Output:** Sentence implementing meaning.

NLG as Planning, Remarks

Historical:

- Long-standing historical connection between NLG and Planning (first mentioned in early 80s).
- Resurrected in 2007, after long silence, thanks to efficiency of heuristic search planners like FF [Hoffmann and Nebel (2001)] → **Chapter 9**.
- Encoding below proposed by [Koller and Stone (2007)].
- Used here in SB at M2CI. (See the "Video Documentary" at `http://www.mmci.uni-saarland.de/en/start`, min. 1:38 – 1:46)

Main advantages of planning in this application:

- Rapid development (try to develop a language generator yourself . . .).
- Flexibility (grammar/knowledge changes handled automatically).

NLG with TAG

NLG in General:

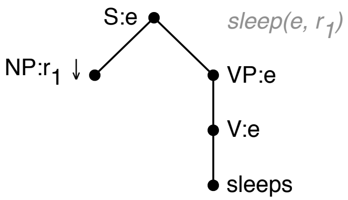
- Given semantic representation (formula) and grammar, compute sentence that expresses this semantics.
- Standard problem in natural language processing, many different approaches exist.

NLG here:

- NLG with tree-adjoining grammars (TAG) [Koller and Stone (2007)].
- Grammar given in form of finite set of elementary trees.
- Problem instance given by grammar, knowledge base, and a set of ground atoms which the sentence should express.

NLG with TAG: Example

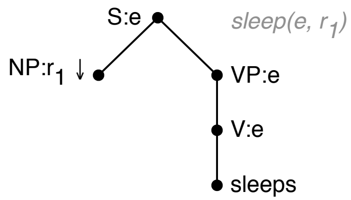
Task: Express ground atom $\{sleep(e, r_1)\}$.
Knowledge Base: $\{sleep(e, r_1),\ rabbit(r_1),\ white(r_1),\ rabbit(r_2)\}$.



- "S:e" stands for sentence referring to event e.
- "NP:r_1 ↓" stands for a noun phrase referring to r_1, which must be substituted here.
- ["VP:e" and "V:e" stand for a verb phrase referring to e, and can be used to adjoin further trees (not detailed here).]

NLG with TAG: Example, ctd.

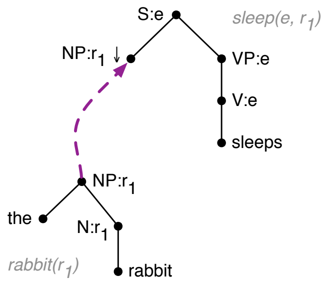
Task: Express ground atom $\{sleep(e, r_1)\}$.
Knowledge Base: $\{sleep(e, r_1),\ rabbit(r_1),\ white(r_1),\ rabbit(r_2)\}$.



- Is this a complete sentence derivation?

NLG with TAG: Example, ctd.

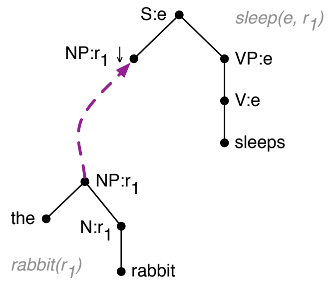
Task: Express ground atom $\{sleep(e, r_1)\}$.
Knowledge Base: $\{sleep(e, r_1),\ rabbit(r_1),\ white(r_1),\ rabbit(r_2)\}$.



- This is a substitution operation (purple dashed arrow in our illustration).
- "N:r_1" stands for a noun-phrase element referring to r_1, and can be used to adjoin further trees.

NLG with TAG: Example, ctd.

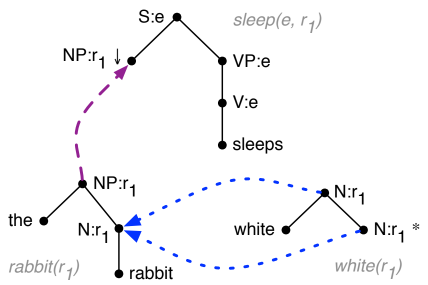
Task: Express ground atom $\{sleep(e, r_1)\}$.
Knowledge Base: $\{sleep(e, r_1),\ rabbit(r_1),\ white(r_1),\ rabbit(r_2)\}$.



- Is this a complete sentence derivation?
- Does the sentence express the desired meaning?

NLG with TAG: Example, ctd.

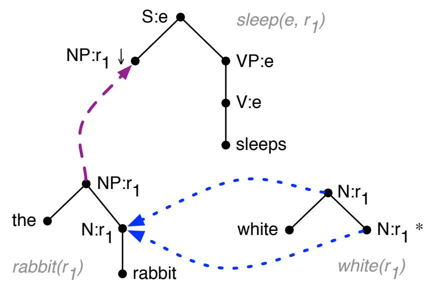
Task: Express ground atom $\{sleep(e, r_1)\}$.
Knowledge Base: $\{sleep(e, r_1),\ rabbit(r_1),\ white(r_1),\ rabbit(r_2)\}$.



- This is an adjunction operation (blue dotted arrow in our illustration).
- "N:r_1" stands for a noun-phrase element referring to r_1, and can be used to adjoin further trees.

NLG with TAG: Example, ctd.

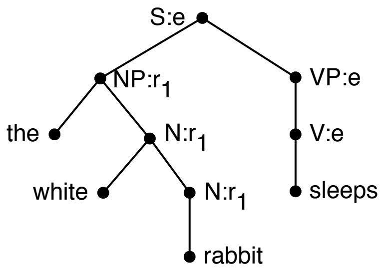
Task: Express ground atom $\{sleep(e, r_1)\}$.
Knowledge Base: $\{sleep(e, r_1),\ rabbit(r_1),\ white(r_1),\ rabbit(r_2)\}$.



- Is this a complete sentence derivation?
- Does the sentence express the desired meaning?

NLG with TAG: Example, ctd.

Task: Express ground atom $\{sleep(e, r_1)\}$.
Knowledge Base: $\{sleep(e, r_1),\ rabbit(r_1),\ white(r_1),\ rabbit(r_2)\}$.



- The outcome of our substitution and adjunction operations here.
- To obtain the desired sentence, read off the leaves from left to right.

... and now in PDDL!

From [Koller and Hoffmann (2010)], slightly simplified:

sleeps(u, u', x, x'):
pre: $subst(S, u),\ ref(u, x),\ sleep(x, x')$
eff: $expressed(sleep, x, x'),\ \neg subst(S, u),$
$subst(NP, u'),\ ref(u', x'),$
$\forall y. y \neq x' \rightarrow distractor(u', y)$

"u, u'": nodes in grammar trees
"x": event
"x'": sentence subject

rabbit(u', x'):
pre: $subst(NP, u'),\ ref(u', x'),\ rabbit(x')$
eff: $\neg subst(NP, u'),\ canadjoin(N, u'),$
$\forall y. \neg rabbit(y) \rightarrow \neg distractor(u', y)$

white(u', x'):
pre: $canadjoin(N, u'),\ ref(u', x'),\ white(x')$
eff: $\forall y. \neg white(y) \rightarrow \neg distractor(u', y)$

Initial state: $subst(S, u_0),\ ref(u_0, e),\ sleep(e, r_1),\ rabbit(r_1), \dots$

Goal: $expressed(sleep, e, r_1)$
$\forall u \forall x. \neg subst(u, x)$
$\forall u \forall x. \neg distractor(u, x)$

Plan: $\langle$**sleeps**(u_0, u_1, e, r_1), **rabbit**(u_1, r_1), **white**$(u_1, r_1)\rangle$.

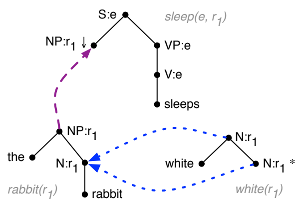
Questionnaire

Question!

In the action "sleeps(u, u', x, x')" in our NLG problem instance, what for do we need the effect literal "$\neg subst(S, u)$"?

(A): So we don't fall asleep.
(B): So the rabbit does not fall asleep.
(C): To mark the subject of S as being open.
(D): To mark S itself as closed.

sleeps(u, u', x, x'):
pre: $subst(S, u),\ ref(u, x),\ sleep(x, x')$
eff: $expressed(sleep, x, x'),\ \neg subst(S, u),$
$subst(NP, u'),\ ref(u', x'),$
$\forall y. y \neq x' \rightarrow distractor(u', y)$



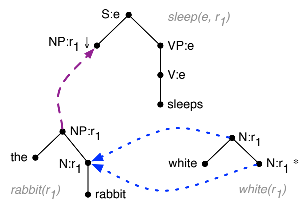
Questionnaire, ctd.

Question!

When we apply the action "sleeps(u, u', e, r_1)" in our NLG problem instance, what does "u'" stand for?

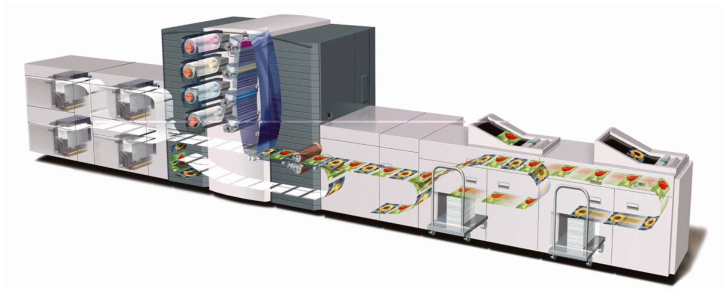
(A): The verb phrase.
(B): The noun phrase.
(C): The node "NP:$r_1 \downarrow$" in the verb-phrase tree.
(D): The tree representing the noun phrase.

sleeps(u, u', x, x'):
pre: $subst(S, u),\ ref(u, x),\ sleep(x, x')$
eff: $expressed(sleep, x, x'),\ \neg subst(S, u),$
$subst(NP, u'),\ ref(u', x'),$
$\forall y. y \neq x' \rightarrow distractor(u', y)$



Research → FAI BSc/MSc/HiWi; Cooperation CoLi

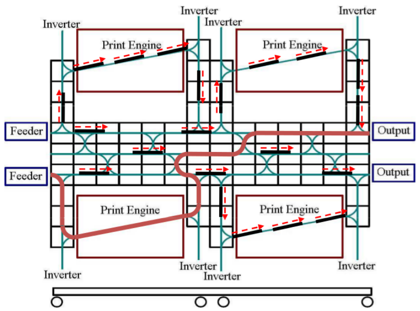
Large-Scale Printing Systems: Complex stuff already . . .



- Process blank sheets of paper into anything (book/bill in folded envelope, . . .).
- Hundreds of independently controlled processing components.
- Dozens of different processes active at any one time.
- Online problem, new jobs come in as we go.

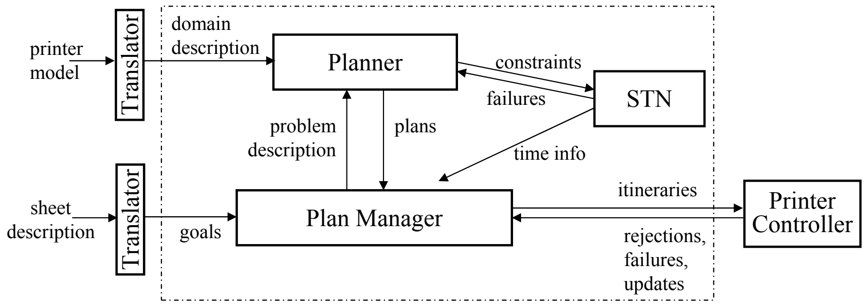
. . . and now we're making it MUCH worse!

MODULAR Large-Scale Printing Systems:



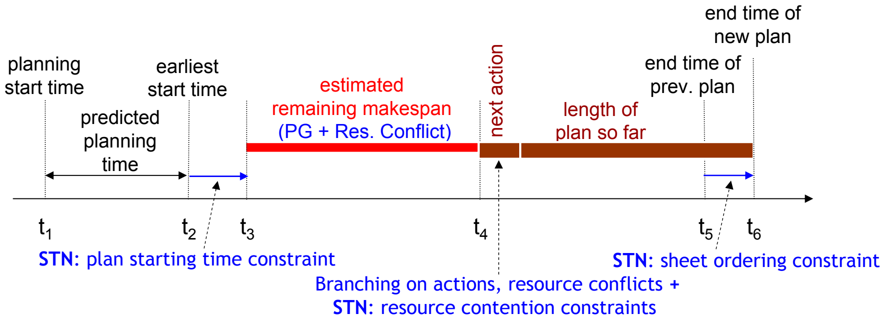
- Assemble and configure components as required by customer.
- No need to buy stuff you don't want, easy to adapt as needed.
- Control can no longer be pre-programmed/configured for a particular machine.
- Requires flexible software that can control anything we could build!

Planning To the Rescue!



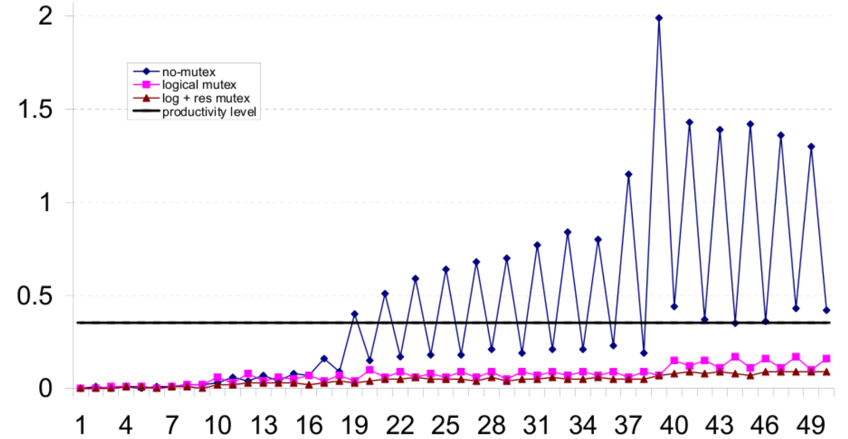
- "Planner" as opposed to "Plan Manager": Finding a solution for the task at any given point in time, vs. managing the updates to the task (new jobs arriving, job cancelled due to paper jam, . . .).
- "STN": Simple Temporal Network. A constraint-based representation of action durations and precedence constraints, identifying unresolvable conflicts.
- The rest should be self-explanatory . . .

Planning To the Rescue! Ctd.



- Regression search → **Chapter 6** using A^* → **Chapter 7**.
- Heuristic function: A temporal variant of h^2 ("PG" here is for "planning graph") → **Chapter 8**.
- "Planning start time", "predicted planning time", "plan starting timing constraint", "resource conflicts", "resource contention constraints", "end time of prev./new plan": Relate to online/temporal aspects of domain.

Empirical Performance



- x-axis: jobs come in during online processing; y-axis: runtime (seconds) for planning the new job; productivity level: runtime needed for practicability.
- "no mutex": without h^2 heuristic function.
  → h^2 is the key element making this work!

Research → FAI BSc/MSc/HiWi; Cooperation DFKI

Summary

- Thanks to the efficiency of heuristic search planning techniques, planning is being applied in a broad variety of applications today.
- Simulated penetration testing is used for regular network security checks, and is commercially employed with FF as the underlying planner.
- Natural language generation involves constructing sentences, and can be successfully encoded into PDDL using FF.
- Flexible printer system control is required for large-scale configurable printing systems, and can be successfully tackled using a temporal variant of the planning heuristic h^2.

Remarks

There's quite a range of **further application areas:**

- Greenhouse logistics involves moving a series of conveyor belts to cater for the needs of all the plants [Helmert and Lasinger (2010)].
- Plan recognition involves observing (some of) the actions of an agent, and inferring what the goal is [Ramírez and Geffner (2009)].
- Business process management involves creating, maintaining, and executing complex processes across large enterprises; planning can be used to automatically generate process templates [Hoffmann *et al.* (2012)].
- Software model checking involves (amongst others) finding bugs; this can be formulated as finding a plan to an error state [Kupferschmid *et al.* (2006)].

Reading

- *Simulated Penetration Testing: From "Dijkstra" to "Turing Test++"* [Hoffmann (2015)].

  Available at:

  http://fai.cs.uni-saarland.de/hoffmann/papers/icaps15inv.pdf

  Content: Overview of simulated pentesting models, systematization of framework with respect to possible models of uncertainty, and with respect to the complexity of the action models considered.

Reading

- *Waking Up a Sleeping Rabbit: On Natural-Language Sentence Generation with FF* [Koller and Hoffmann (2010)].

  Available at:

  http://fai.cs.uni-saarland.de/hoffmann/papers/icaps10.pdf

  Content: Summarizes the NLG problem based on TAG, and its encoding into PDDL. Gives a compact summary of the problems initially encountered with off-the-shelf FF, and the minor fixes required to get rid of those problems; runs experiments showing the dramatic performance gains obtained this way, making this approach practical. Discusses open issues for planning technology in this domain.

Reading

- *On-line Planning and Scheduling: An Application to Controlling Modular Printers* [Ruml *et al.* (2011)].

  Available at:

  http://www.jair.org/media/3184/live-3184-5462-jair.pdf

  Content: Comprehensive and detailed description of the application context, the configuration of planning and scheduling techniques used, and the added value obtained in doing so.

  For a shorter introduction of this application, refer to [Ruml *et al.* (2005)] available at:

  http://www.cs.unh.edu/~ruml/papers/icaps-05-revised-1.pdf

References I

Mark Boddy, Jonathan Gohde, Tom Haigh, and Steven Harp. Course of action generation for cyber security using classical planning. In Susanne Biundo, Karen Myers, and Kanna Rajan, editors, *Proceedings of the 15th International Conference on Automated Planning and Scheduling (ICAPS-05)*, pages 12–21, Monterey, CA, USA, 2005. Morgan Kaufmann.

Malte Helmert and Hauke Lasinger. The Scanalyzer domain: Greenhouse logistics as a planning problem. In Ronen I. Brafman, Hector Geffner, Jörg Hoffmann, and Henry A. Kautz, editors, *Proceedings of the 20th International Conference on Automated Planning and Scheduling (ICAPS'10)*, pages 234–237. AAAI Press, 2010.

Jörg Hoffmann and Bernhard Nebel. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, 14:253–302, 2001.

Jörg Hoffmann, Ingo Weber, and Frank Michael Kraft. SAP speaks PDDL: Exploiting a software-engineering model for planning in business process management. *Journal of Artificial Intelligence Research*, 44:587–632, 2012.

References II

Jörg Hoffmann. Simulated penetration testing: From "Dijkstra" to "Turing Test++". In Ronen Brafman, Carmel Domshlak, Patrik Haslum, and Shlomo Zilberstein, editors, *Proceedings of the 25th International Conference on Automated Planning and Scheduling (ICAPS'15)*. AAAI Press, 2015.

Alexander Koller and Jörg Hoffmann. Waking up a sleeping rabbit: On natural-language sentence generation with ff. In Ronen I. Brafman, Hector Geffner, Jörg Hoffmann, and Henry A. Kautz, editors, *Proceedings of the 20th International Conference on Automated Planning and Scheduling (ICAPS'10)*. AAAI Press, 2010.

Alexander Koller and Matthew Stone. Sentence generation as planning. In *Proc. of the 45th Annual Meeting of the Association for Computational Linguistics (ACL'07)*, 2007.

Sebastian Kupferschmid, Jörg Hoffmann, Henning Dierks, and Gerd Behrmann. Adapting an AI planning heuristic for directed model checking. In Antti Valmari, editor, *Proceedings of the 13th International SPIN Workshop (SPIN 2006)*, volume 3925 of *Lecture Notes in Computer Science*, pages 35–52. Springer-Verlag, 2006.

Jorge Lucangeli, Carlos Sarraute, and Gerardo Richarte. Attack planning in the real world. In *Proceedings of the 2nd Workshop on Intelligent Security (SecArt'10)*, 2010.

References III

Miquel Ramírez and Hector Geffner. Plan recognition as planning. In Craig Boutilier, editor, *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI'09)*, pages 1778–1783, Pasadena, California, USA, July 2009. Morgan Kaufmann.

Wheeler Ruml, Minh Do, and Markus Fromherz. On-line planning and scheduling for high-speed manufacturing. In Susanne Biundo, Karen Myers, and Kanna Rajan, editors, *Proceedings of the 15th International Conference on Automated Planning and Scheduling (ICAPS-05)*, pages 30–39, Monterey, CA, USA, 2005. Morgan Kaufmann.

Wheeler Ruml, Minh Binh Do, Rong Zhou, and Markus P. J. Fromherz. On-line planning and scheduling: An application to controlling modular printers. *Journal of Artificial Intelligence Research*, 40:415–468, 2011.