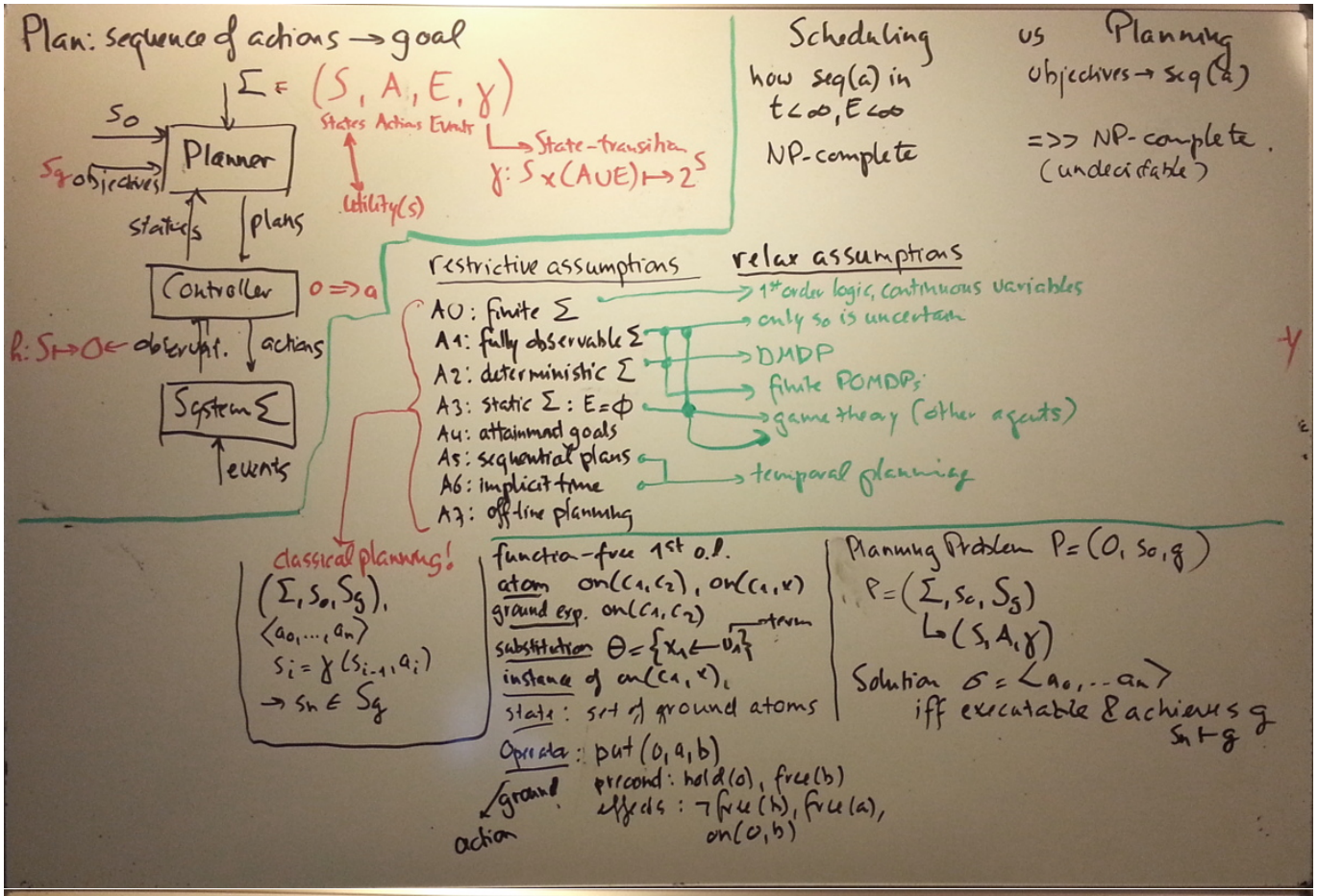
Summary Planning and Scheduling

Alexander Hagg
Bonn-Rhein-Sieg University of Applied Sciences
Master Course on Autonomous Systems
Grantham-Allee 20, 53757 Sankt Augustin
Germany

Sankt Augustin, Decmber 5, 2013


Plan: sequence of actions → goal
Σ = (S, A, E, γ)
States Actions Events
State-transition
γ: S × (A ∪ E) → 2^S
Utility(s)
s0
sg objectives
Planner
states
plans
Controller
o ⇒ a
h: S → O
observat.
actions
System Σ
events
Scheduling
how seq(a) in
t < ∞, E < ∞
NP-complete
vs Planning
objectives → seq(a)
⇒ NP-complete.
(undecidable)
restrictive assumptions
relax assumptions
A0: finite Σ
A1: fully observable Σ
A2: deterministic Σ
A3: static Σ : E = ∅
A4: attainment goals
A5: sequential plans
A6: implicit time
A7: off-line planning
1st order logic, continuous variables
only so is uncertain
DMDP
finite POMDPs
game theory (other agents)
temporal planning
classical planning!
(Σ, s0, Sg),
⟨a0, ..., an⟩
si = γ(si-1, ai)
→ sn ∈ Sg
function-free 1st o.l.
atom on(c1, c2), on(c1, x)
ground exp. on(c1, c2)
substitution θ = {x1 ← u1}
term
instance of on(c1, x),
state: set of ground atoms
Operator: put(o, a, b)
precond: hold(o), free(b)
effects: ¬free(h), free(a),
on(o, b)
ground action
Planning Problem P = (O, s0, g)
P = (Σ, s0, Sg)
↳ (S, A, γ)
Solution σ = ⟨a0, .. an⟩
iff executable & achieves g
sn ⊢ g

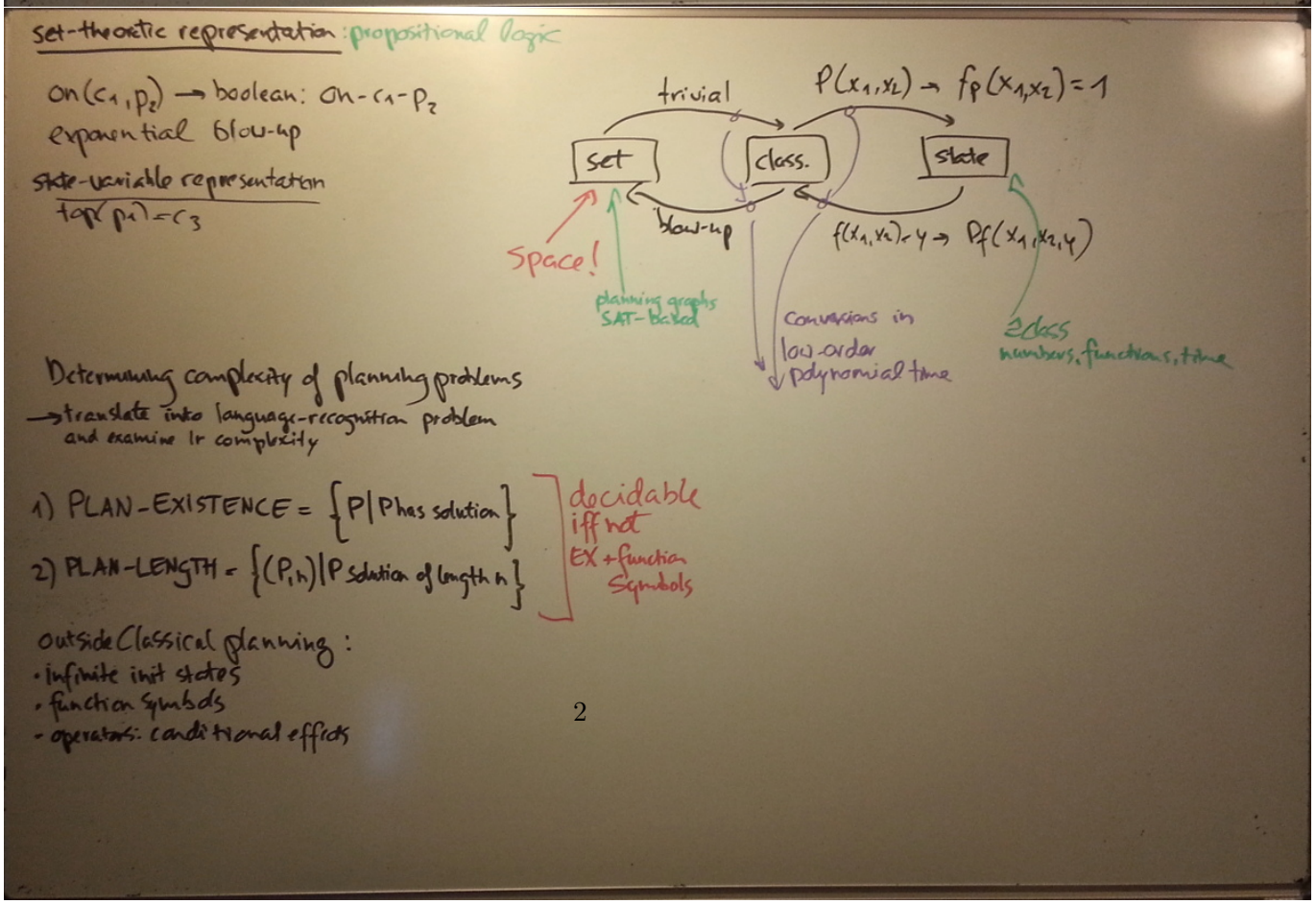

set-theoretic representation: propositional logic
on(c1, p2) → boolean: on-c1-p2
exponential blow-up
state-variable representation
top(p1) = c3
trivial
P(x1, x2) → fp(x1, x2) = 1
set
class.
state
blow-up
f(x1, x2) = y → Pf(x1, x2, y)
Space!
planning graphs
SAT-based
conversions in
low-order
polynomial time
2 class
numbers, functions, time
Determining complexity of planning problems
→ translate into language-recognition problem
and examine its complexity
1) PLAN-EXISTENCE = {P | P has solution}
2) PLAN-LENGTH = {(P, n) | P solution of length n}
decidable
iff not
EX + function
symbols
Outside Classical planning:
• infinite init states
• function symbols
• operators: conditional effects

Figure 1:

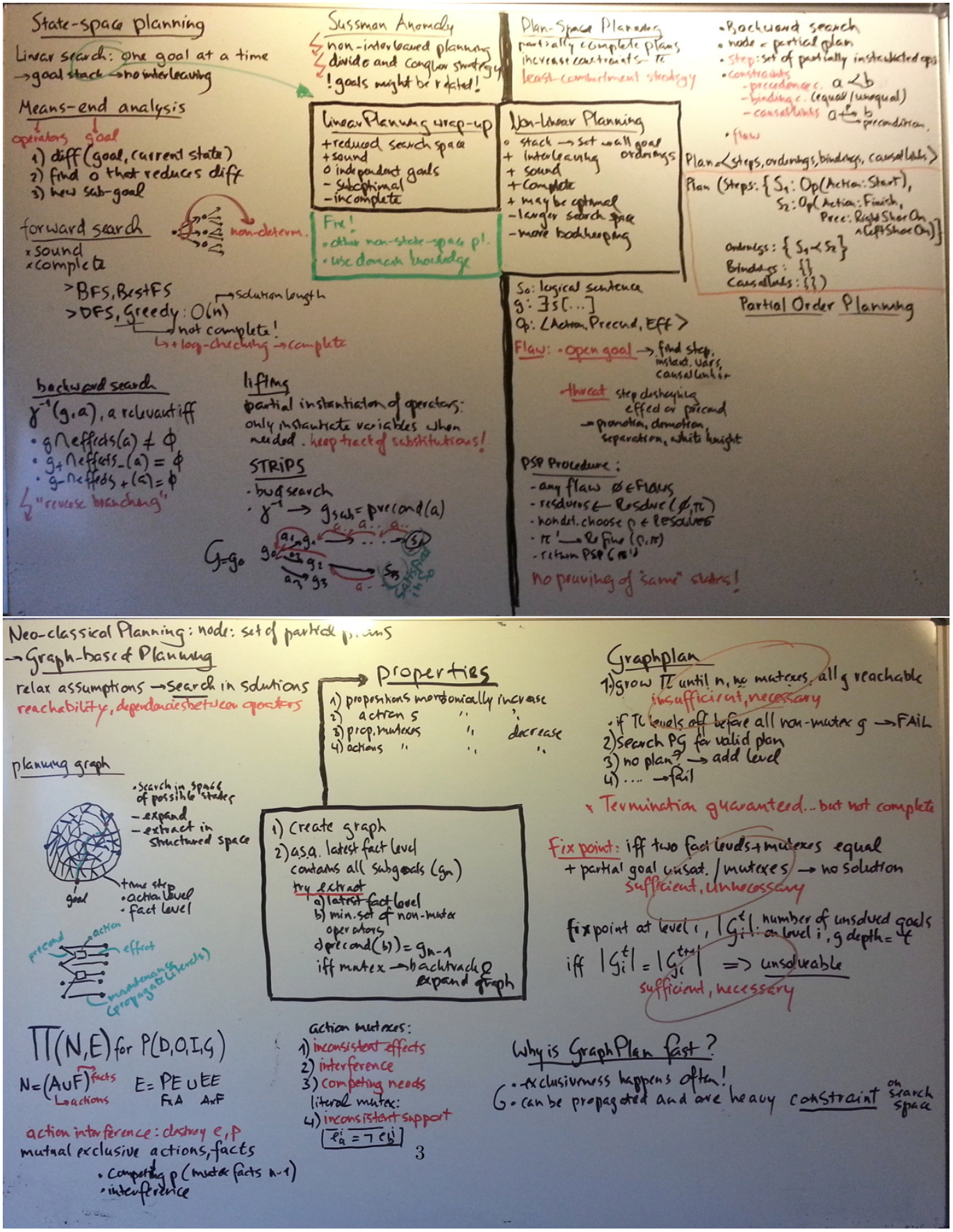

State-space planning
Linear search: one goal at a time
→goal stack →no interleaving
Means-end analysis
operators goal
1) diff(goal, current state)
2) find o that reduces diff
3) new sub-goal
forward search
sound
complete
non-determ.
>BFS, BestFS
>DFS, Greedy: O(n)
solution length
not complete!
+ loop-checking →complete
backward search
γ⁻¹(g,a), a relevant iff
• g ∩ effects(a) ≠ ϕ
• g+ ∩ effects-(a) = ϕ
• g- ∩ effects+(a) = ϕ
"reverse branching"
lifting
partial instantiation of operators:
only instantiate variables when needed. keep track of substitutions!
STRIPS
- bwd search
- γ⁻¹ → g_sub = precond(a)
G=g0
Sussman Anomaly
non-interleaved planning
divide and conquer strategy
! goals might be related!
Linear Planning wrap-up
+reduced search space
+sound
0 independent goals
- suboptimal
-incomplete
Fix!
• other non-state-space p!
• use domain knowledge
Non-Linear Planning
o stack → set of all goal orderings
+ interleaving
+ sound
+ Complete
+ may be optimal
-larger search space
-more bookkeeping
So: logical sentence
g: ∃s[...]
Op: ⟨Action, Precond, Eff⟩
Flaw: • open goal → find step, instant. vars, causal link
• threat step clobbering effect or precond
→ promotion, demotion, separation, white knight
PSP Procedure:
- any flaw ∅ ∈ Flaws
- resolvers ← Resolve(∅, π)
- nondet. choose ρ ∈ Resolvers
- π' → Refine(ρ, π)
- return PSP(π')
no pruning of "same" states!
Plan-Space Planning
partially complete plans
increase constraints - TE
least-commitment strategy
• backward search
• node = partial plan
• step: set of partially instantiated ops
• constraints
- precedence c: a ≺ b
- binding c: (equal / unequal)
- causal links a →p b (precondition)
• flaw
Plan ⟨steps, orderings, bindings, causal links⟩
Plan (Steps: {S1: Op(Action: Start), S2: Op(Action: Finish, Prec: RightShoeOn ∧ LeftShoeOn)}
Orderings: {S1 ≺ S2}
Bindings: {}
Causal links: {})
Partial Order Planning
Neo-classical Planning: node: set of partial plans
→Graph-based Planning
relax assumptions →search in solutions
reachability, dependencies between operators
properties
1) propositions monotonically increase
2) actions " "
3) prop. mutexes " decrease
4) actions " "
planning graph
• search in space of possible states
- expand
- extract in structured space
goal
time step
• action level
• fact level
precond
action
effect
maintenance (propagate literals)
1) create graph
2) as.a. latest fact level contains all subgoals (gn)
try extract
a) latest fact level
b) min. set of non-mutex operators
precond(b)) = gn-1
iff mutex →backtrack & expand graph
Graphplan
1) grow Π until n, no mutexes, all g reachable
insufficient, necessary
• if Π levels off before all non-mutex g →FAIL
2) search PG for valid plan
3) no plan? → add level
4) .... →fail
x Termination guaranteed... but not complete
Fix point: iff two fact levels + mutexes equal
+ partial goal unsat. / mutexes → no solution
sufficient, unnecessary
fixpoint at level i, |G_i^t|: number of unsolved goals on level i, g depth = t
iff |G_i^t| = |G_i^{t+1}| => unsolvable
sufficient, necessary
Π(N,E) for P(D,O,I,G)
N = (A ∪ F) facts
actions
E = PE ∪ EE
F×A A×F
action interference: destroy e, p
mutual exclusive actions, facts
• competing p (mutex facts n-1)
• interference
action mutexes:
1) inconsistent effects
2) interference
3) competing needs
literal mutex:
4) inconsistent support
[p_a^i = ¬c_b^i]
Why is GraphPlan fast?
• exclusiveness happens often!
G • can be propagated and are heavy constraint on search space

Figure 2: ..