Data Structures for Efficient Inference and Optimization

in Expressive Continuous Domains

Scott Sanner

NICTA

Ehsan Abbasnejad
Zahra Zamani

Karina Valdivia Delgado
Leliane Nunes de Barros

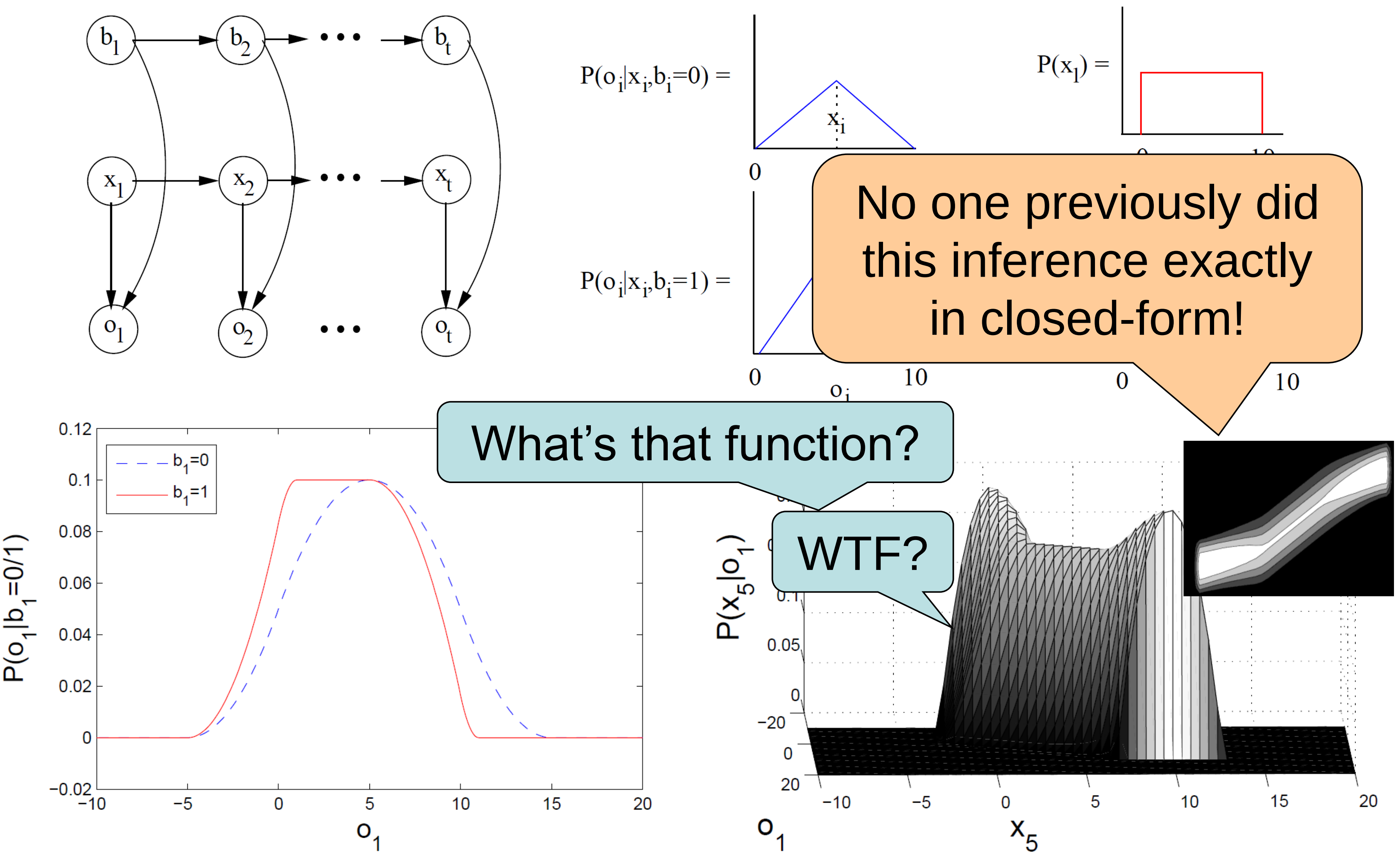
Cheng Fang

THE AUSTRALIAN NATIONAL UNIVERSITY

Universidade de São Paulo

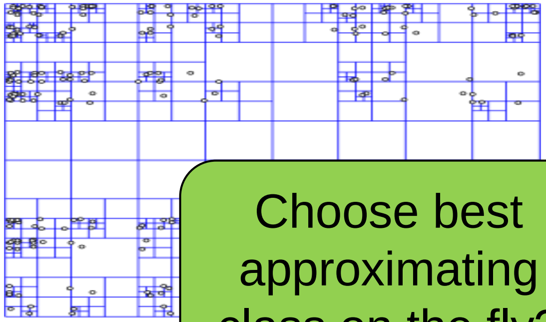
Massachusetts Institute of Technology

Inference for Continuous HMMs



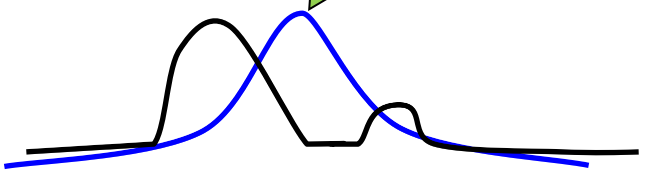
Continuous Inference Solved?

- How is inference done in piecewise models?
  - (Adaptively) discretize model:
    - Approximate, $O(N^D)$
    - Adaptivity is an artform



Choose best approximating class on the fly?

  - Projective approximation: variational, EP
    - Choose approximating class *a priori*
    - Often Gaussian – best choice?



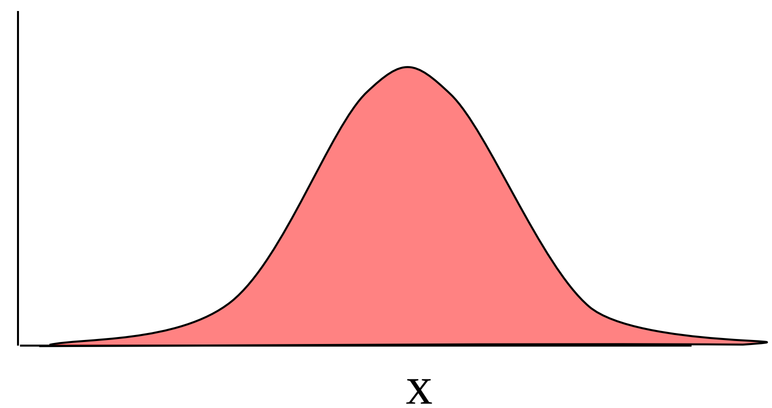
  - Sampling: Monte Carlo, Gibbs
    - May not converge for (near) deterministic distributions
    - Not for evidence as a free variable – E[X | O=?]

What has everyone been missing?

Symbolic representations and operations on piecewise functions

General Form for Continuous Distributions?

- Probability density functions (pdfs), e.g.
  - $N(x; \mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{\sigma^2}}$



- Could be **piecewise** or **deterministic**
  - Stochastic programs (conditionals)
  - Utilities (step), decision-making (max), preferences ($\geq$)
  - Dynamical controlled systems (switching control)
  - Deterministic (δ)

Piecewise Functions (Cases)

$$z = f(x,y) = \begin{cases} \boxed{(x > 3) \wedge (y \cdot \ x) : \quad x + y} \\ (x \cdot \ 3) \vee (y > x) : \quad x^2 + xy^3 \end{cases}$$

Partition

Constraint

Value

Linear constraints and value

Linear constraints, constant value

Quadratic constraints and value

Formal Problem Statement

- General continuous graphical models represented by piecewise functions (cases)

$$f = \begin{cases} \phi_1 : & f_1 \\ \vdots & \vdots \\ \phi_k : & f_k \end{cases}$$

- Exact closed-form solution inferred via the following piecewise calculus:
  - $f_1 \oplus f_2$, $f_1 \otimes f_2$
  - $\max(f_1, f_2)$, $\min(f_1, f_2)$
  - $\int_x f(x)$
  - $\max_x f(x)$, $\min_x f(x)$

Question: how do we perform these operations in closed-form?

Polynomial Case Operations: ⊕, ⊗

$$\begin{cases} \phi_1 : & f_1 \\ \phi_2 : & f_2 \end{cases} \oplus \begin{cases} \psi_1 : & g_1 \\ \psi_2 : & g_2 \end{cases} = \qquad ?$$

Polynomial Case Operations: ⊕, ⊗

$$\begin{cases} \phi_1 : & f_1 \\ \phi_2 : & f_2 \end{cases} \oplus \begin{cases} \psi_1 : & g_1 \\ \psi_2 : & g_2 \end{cases} = \begin{cases} \phi_1 \wedge \psi_1 : & f_1 + g_1 \\ \phi_1 \wedge \psi_2 : & f_1 + g_2 \\ \phi_2 \wedge \psi_1 : & f_2 + g_1 \\ \phi_2 \wedge \psi_2 : & f_2 + g_2 \end{cases}$$

- Similarly for ⊗
  - Polynomials closed under +, *
- What about max?
  - Max of polynomials is not a polynomial ☹

Polynomial Case Operations: max

$$\max\left(\begin{cases} \phi_1 : & f_1 \\ \phi_2 : & f_2 \end{cases}, \begin{cases} \psi_1 : & g_1 \\ \psi_2 : & g_2 \end{cases} \right) =$$

?

Polynomial Case Operations: max

$$\max\left(\begin{cases}\phi_1: & f_1\\ \phi_2: & f_2\end{cases},\begin{cases}\psi_1: & g_1\\ \psi_2: & g_2\end{cases}\right)=\begin{cases}\phi_1\wedge\psi_1\wedge f_1>g_1: & f_1\\ \phi_1\wedge\psi_1\wedge f_1\cdot\ g_1: & g_1\\ \phi_1\wedge\psi_2\wedge f_1>g_2: & f_1\\ \phi_1\wedge\psi_2\wedge f_1\cdot\ g_2: & g_2\\ \phi_2\wedge\psi_1\wedge f_2>g_1: & f_2\\ \phi_2\wedge\psi_1\wedge f_2\cdot\ g_1: & g_1\\ \phi_2\wedge\psi_2\wedge f_2>g_2: & f_2\\ \phi_2\wedge\psi_2\wedge f_2\cdot\ g_2: & g_2\end{cases}$$

- Still a piecewise polynomial!

Size blowup?
We'll get to that…

Definite Integration: $\int_{x=-\infty}^{\infty}$

- Closed for polynomials
  - But how to compute for case?

$$\int_{x=-\infty}^{\infty} \begin{cases} \phi_1 : & f_1 \\ \vdots & \vdots \\ \phi_k : & f_k \end{cases} dx$$

  - Just integrate case partitions, $\oplus$ results!

Partition Integral

1. Determine integration bounds

$$\int_{x=-\infty}^{\infty} [\phi_1] \cdot f_1 \, dx$$

$$\phi_1 := [x > -1] \wedge [x > y - 1] \wedge [x \cdot \ z] \wedge [x \cdot \ y + 1] \wedge [y > 0]$$

$$f_1 := x^2 - xy$$

UB and LB are symbolic!

What constraints here?

- independent of x
- pairwise UB > LB

How to evaluate?

Definite Integral Evaluation

- How to evaluate integral bounds?

$$\int_{x=LB}^{UB} x^2 - xy \;=\; \frac{1}{3}x^3 - \frac{1}{2}x^2 y \Big|_{LB}^{UB}$$

- Can do polynomial operations on cases!

Symbolically, exactly evaluated!

Exact Graphical Model Inference!

(directed and undirected)

- Can do general probabilistic inference

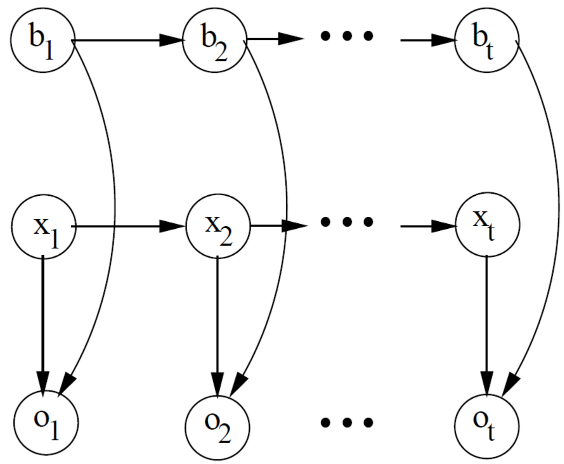
$$p(x_2|x_1) = \frac{\int_{x_3} \cdots \int_{x_n} \bigotimes_{i=1}^{k} case_i \; dx_n \cdots dx_3}{\int_{x_2} \cdots \int_{x_n} \bigotimes_{i=1}^{k} case_i \; dx_n \cdots dx_2}$$

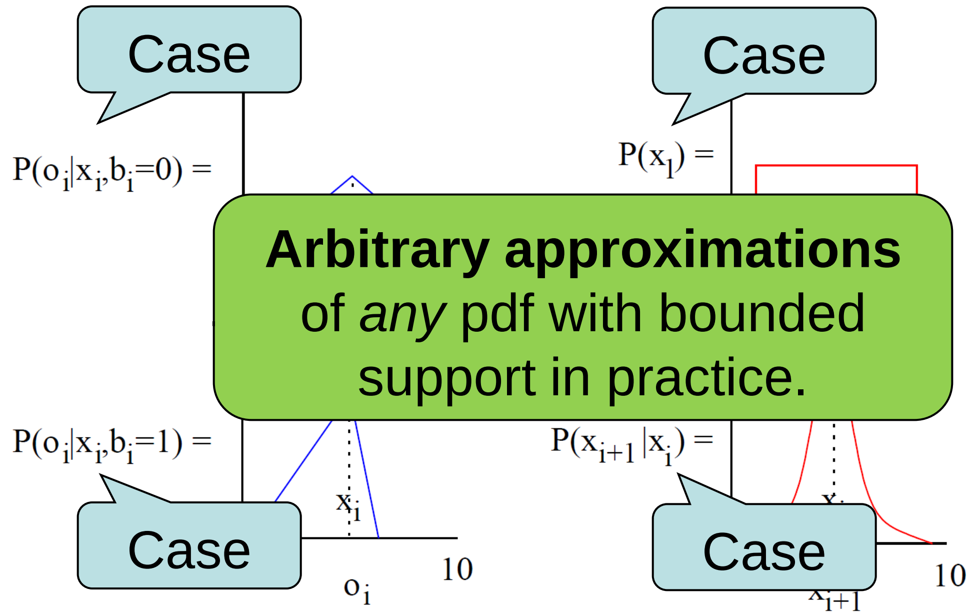
- Or an exact expectation of *any* polynomial

  – *poly* = Mean, variance, skew, curtosis, …, x^2+y^2+xy

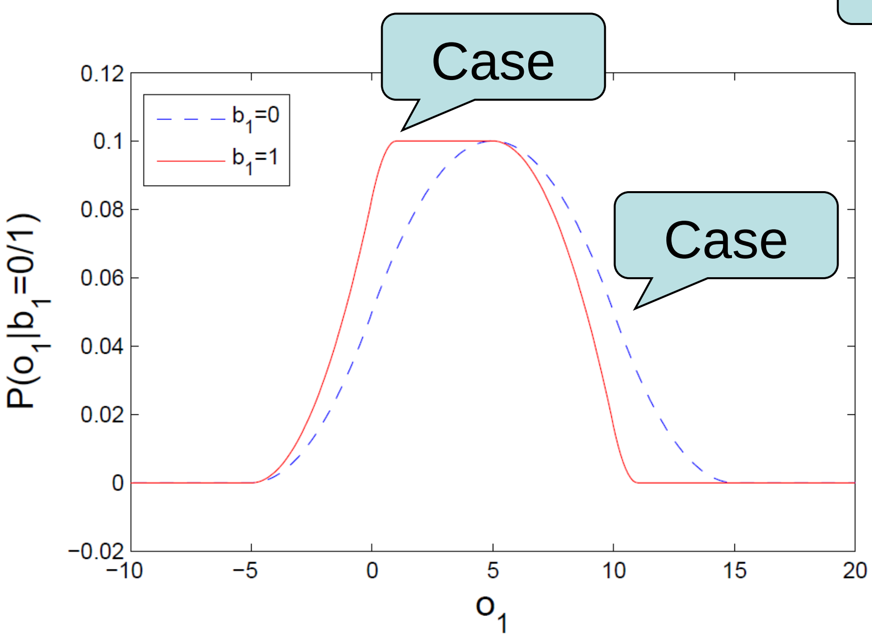
All computed by **Symbolic Variable Elimination (SVE)**

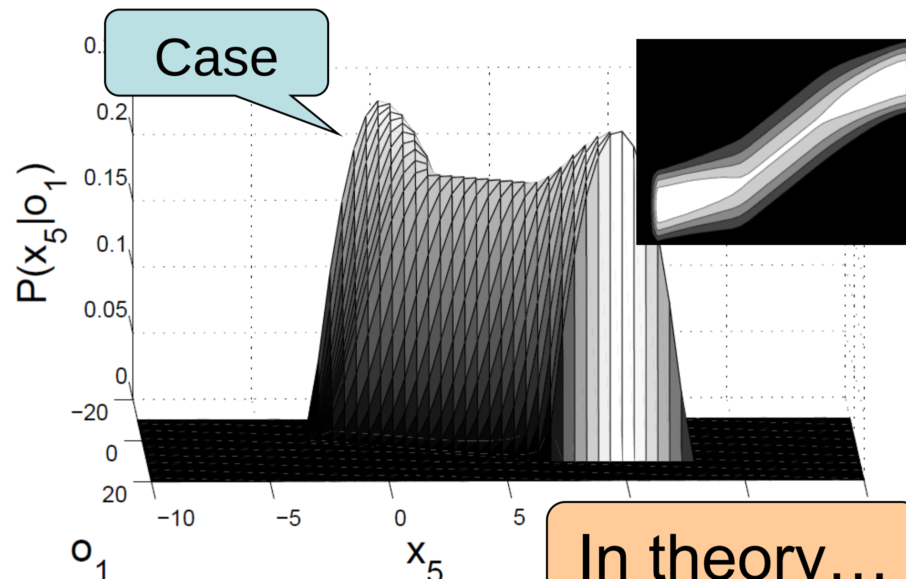
Voila: Closed-form Exact Inference via SVE!





Arbitrary approximations of *any* pdf with bounded support in practice.





In theory…

Computational Complexity?

- In theory for SVE on graphical models
  - Best-case complexity Ω(#operations)
  - Worst-case complexity is O(exp(#operations))
    - Not explicitly tree-width dependent!
    - **But worse:** integral may invoke 100's of operations!

Fortunately decision diagrams can mitigate worst-case complexity

BDD / ADDs

Quick Introduction

Function Representation (Tables)

- How to represent functions: $B^n \rightarrow R$?
- How about a fully enumerated table…
- …OK, but can we be more compact?

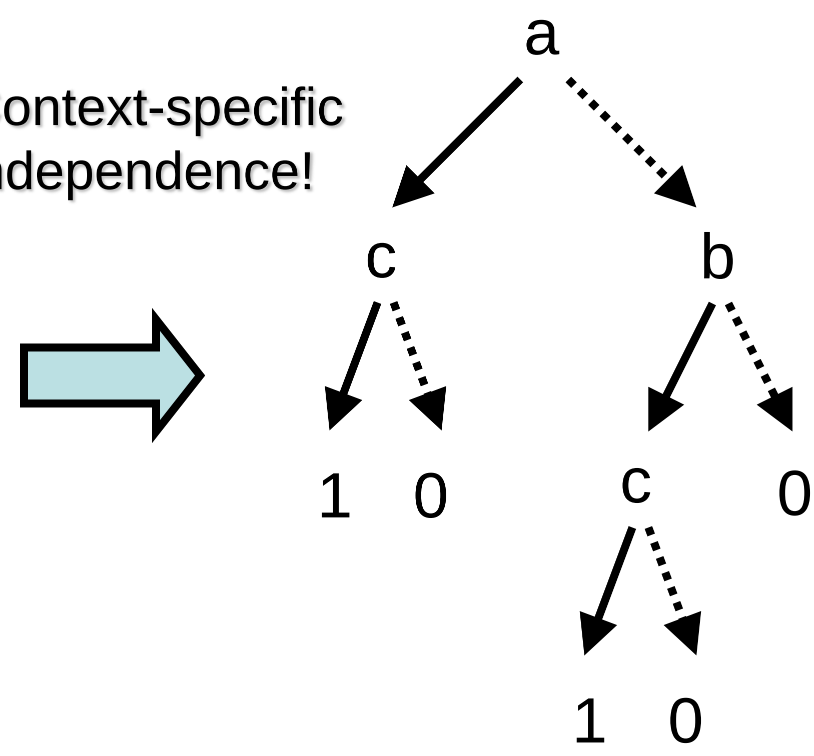
a	b	c	F(a,b,c)
0	0	0	0.00
0	0	1	0.00
0	1	0	0.00
0	1	1	1.00
1	0	0	0.00
1	0	1	1.00
1	1	0	0.00
1	1	1	1.00

Function Representation (Trees)

- How about a tree? Sure, can simplify.

a	b	c	F(a,b,c)
0	0	0	0.00
0	0	1	0.00
0	1	0	0.00
0	1	1	1.00
1	0	0	0.00
1	0	1	1.00
1	1	0	0.00
1	1	1	1.00

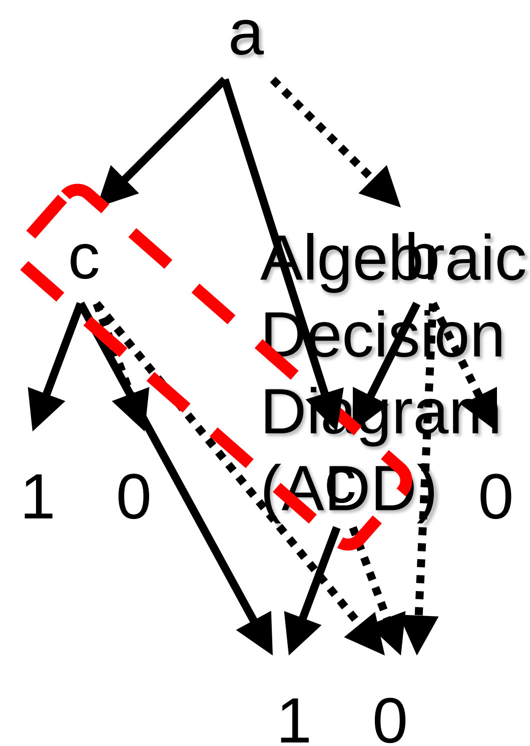
Context-specific independence!



Function Representation (ADDs)

- Why not a directed acyclic graph (DAG)?

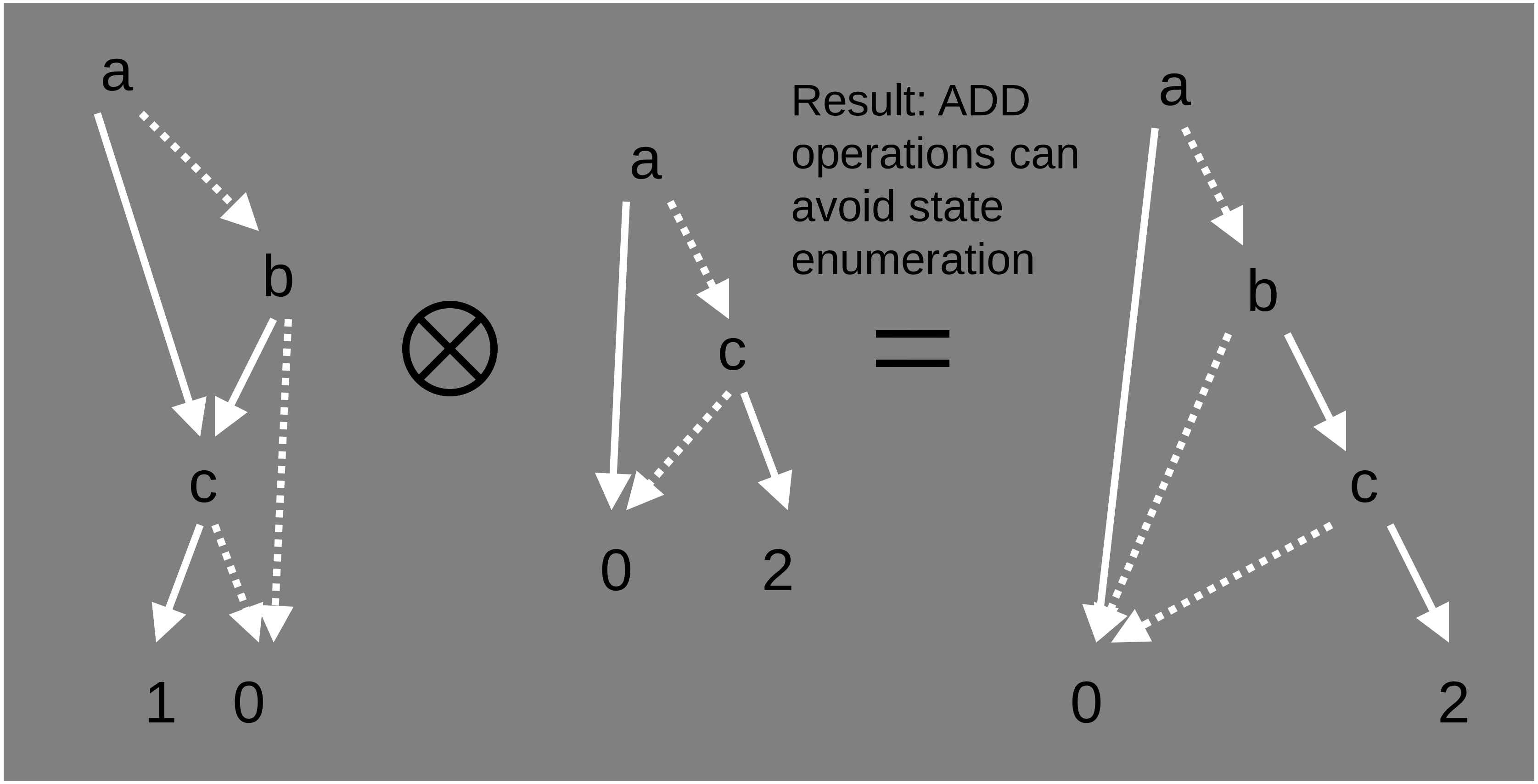
a	b	c	F(a,b,c)
0	0	0	0.00
0	0	1	0.00
0	1	0	0.00
0	1	1	1.00
1	0	0	0.00
1	0	1	1.00
1	1	0	0.00
1	1	1	1.00



Exploits context-specific independence (CSI) and shared substructure.

Binary Operations (ADDs)

- Why do we order variable tests?
- Enables us to do efficient binary operations…



Case → XADD

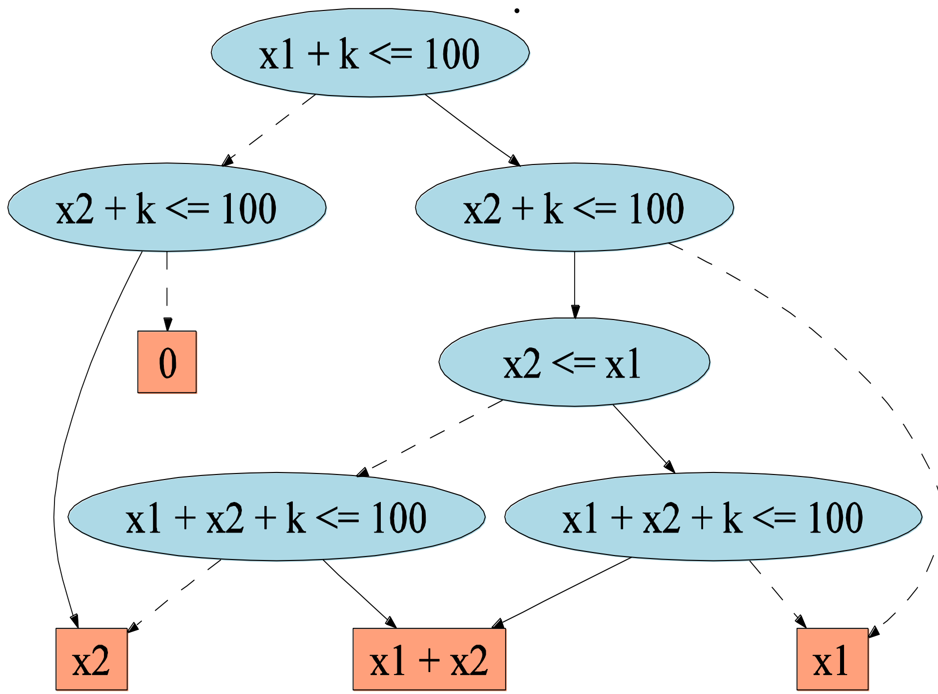
XADD = continuous variable e**x**tension
of **a**lgebraic **d**ecision **d**iagram

Efficient XADD data structure for cases
• strict ordering of atomic inequality tests

➔ compact, minimal case representation
➔ efficient case operations

Case → XADD

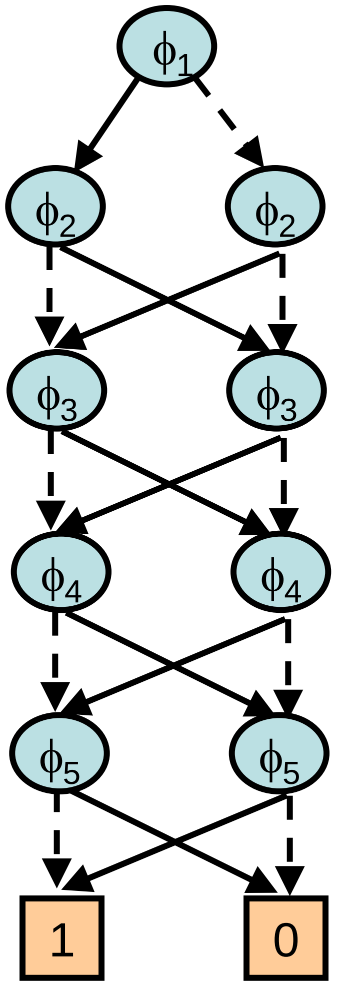
$$V = \begin{cases} x_1 + k > 100 \wedge x_2 + k > 100 : & 0 \\ x_1 + k > 100 \wedge x_2 + k \leq 100 : & x_2 \\ x_1 + k \leq 100 \wedge x_2 + k > 100 : & x_1 \\ x_1 + x_2 + k > 100 \wedge x_1 + k \leq 100 \wedge x_2 + k \leq 100 \wedge x_2 > x_1 : & x_2 \\ x_1 + x_2 + k > 100 \wedge x_1 + k \leq 100 \wedge x_2 + k \leq 100 \wedge x_2 \leq x_1 : & x_1 \\ x_1 + x_2 + k \leq 100 : & x_1 + x_2 \\ \vdots & \vdots \end{cases}$$



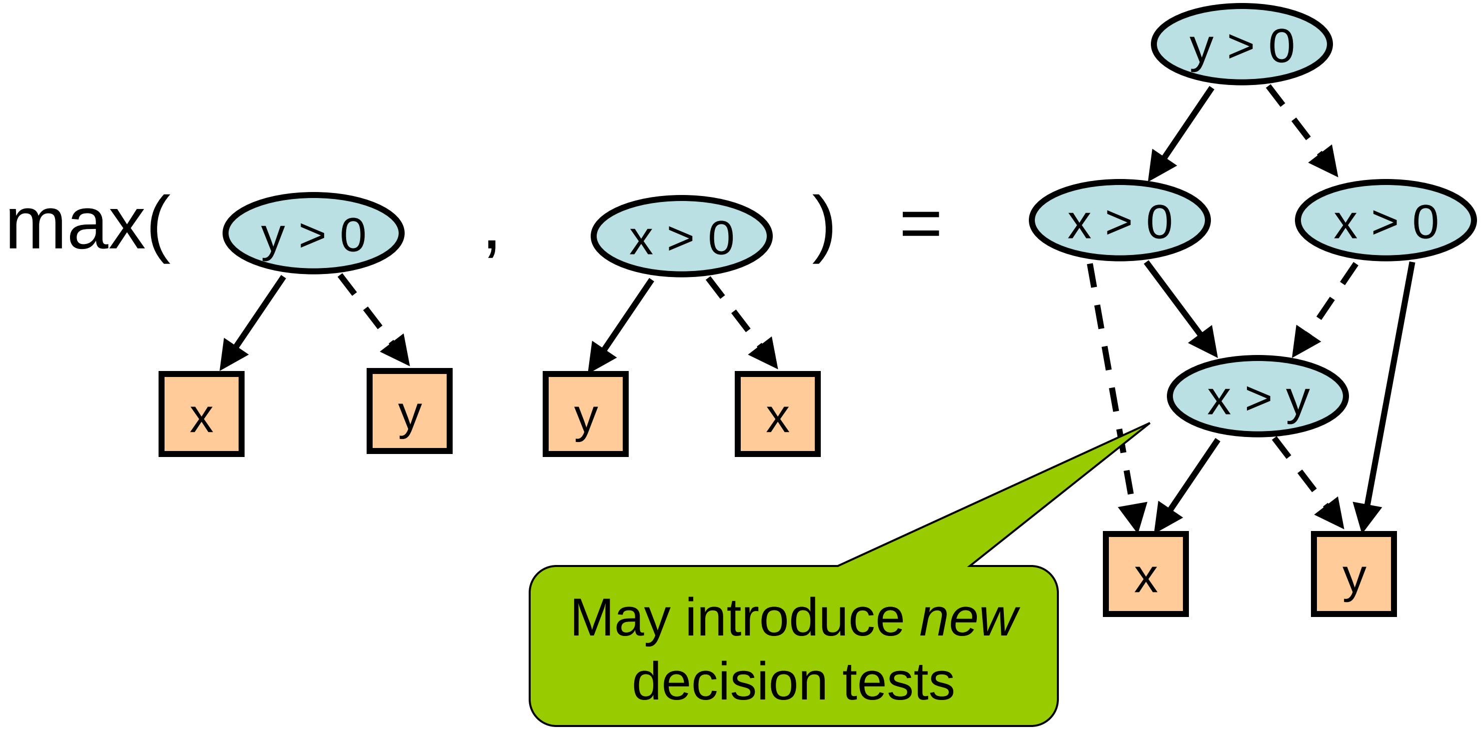
*With non-trivial extensions over ADD, can reduce to a minimal canonical form!

Compactness of (X)ADDs

- Linear in number of decisions ϕ_i
- Case version has exponential number of partitions!



XADD Maximization



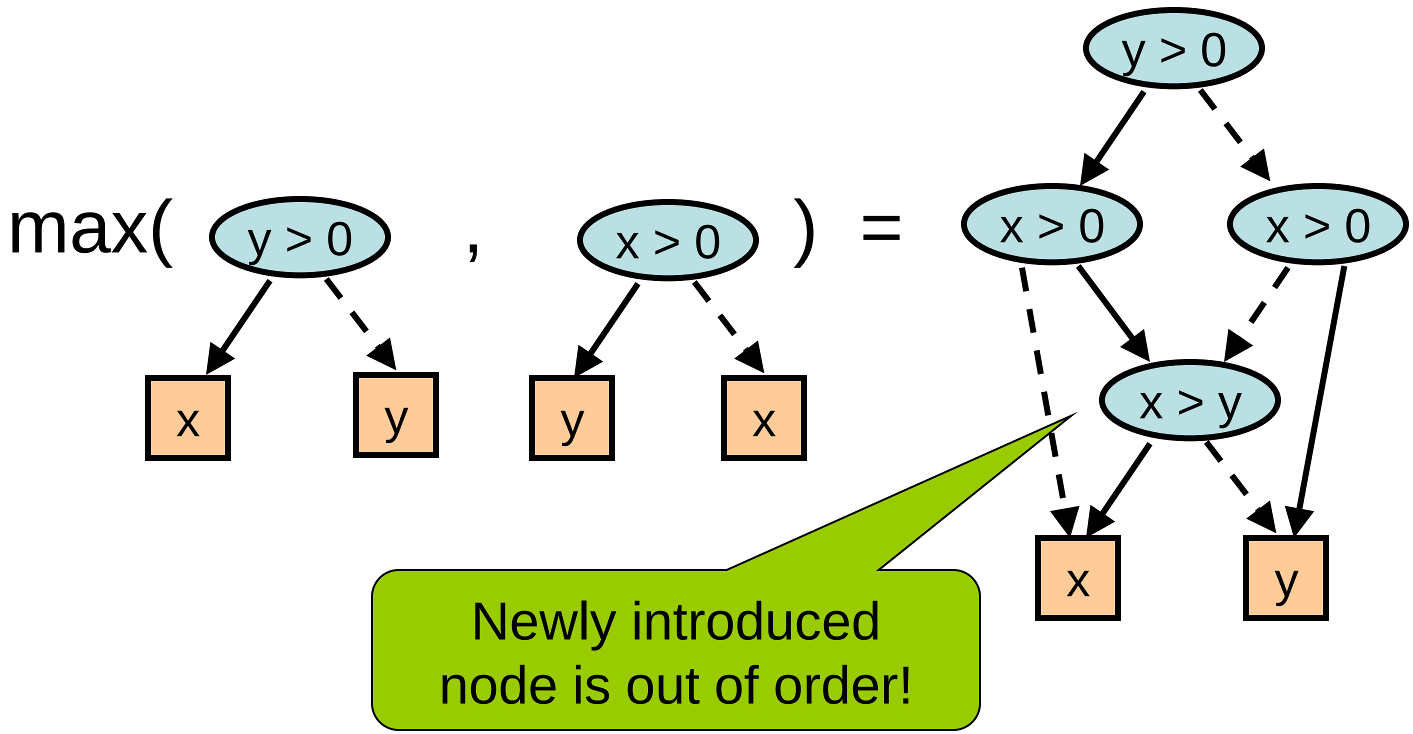
Operations exploit structure: O(|f||g|)

Maintaining XADD Orderings

- Max may get decisions out of order

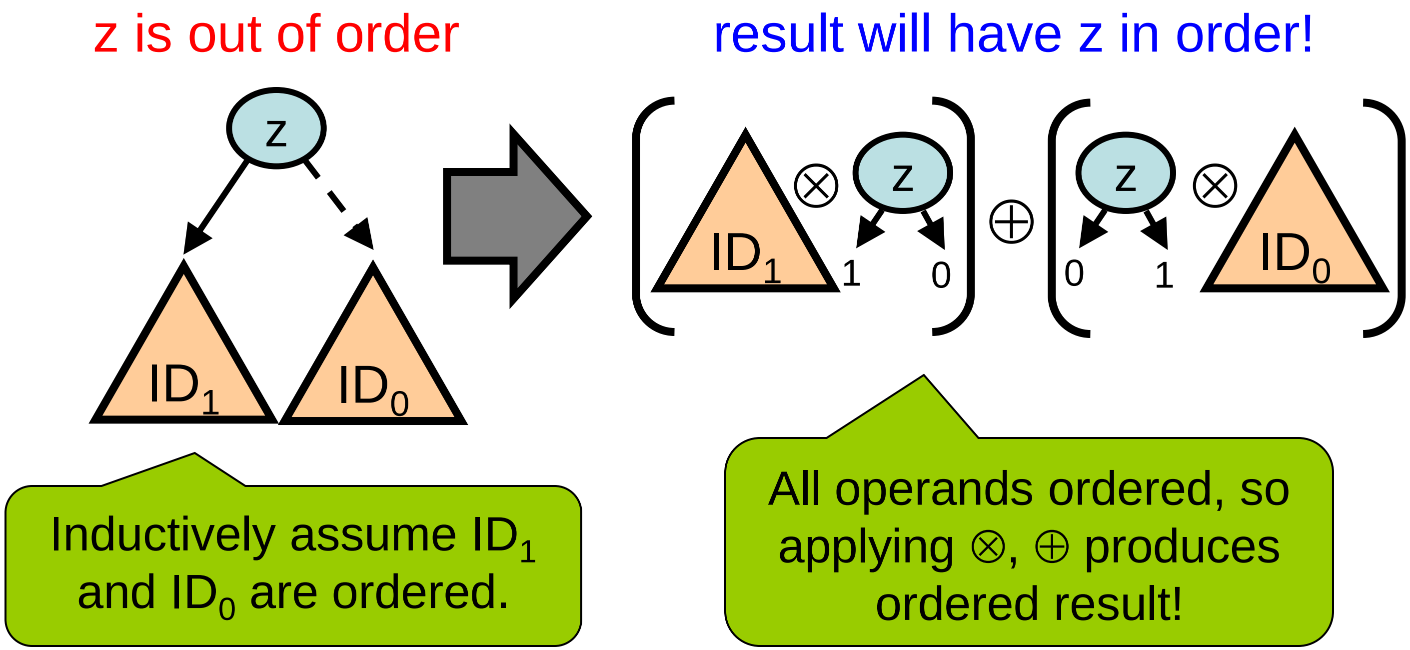
Decision ordering (root→leaf)

- x > y
- y > 0
- x > 0

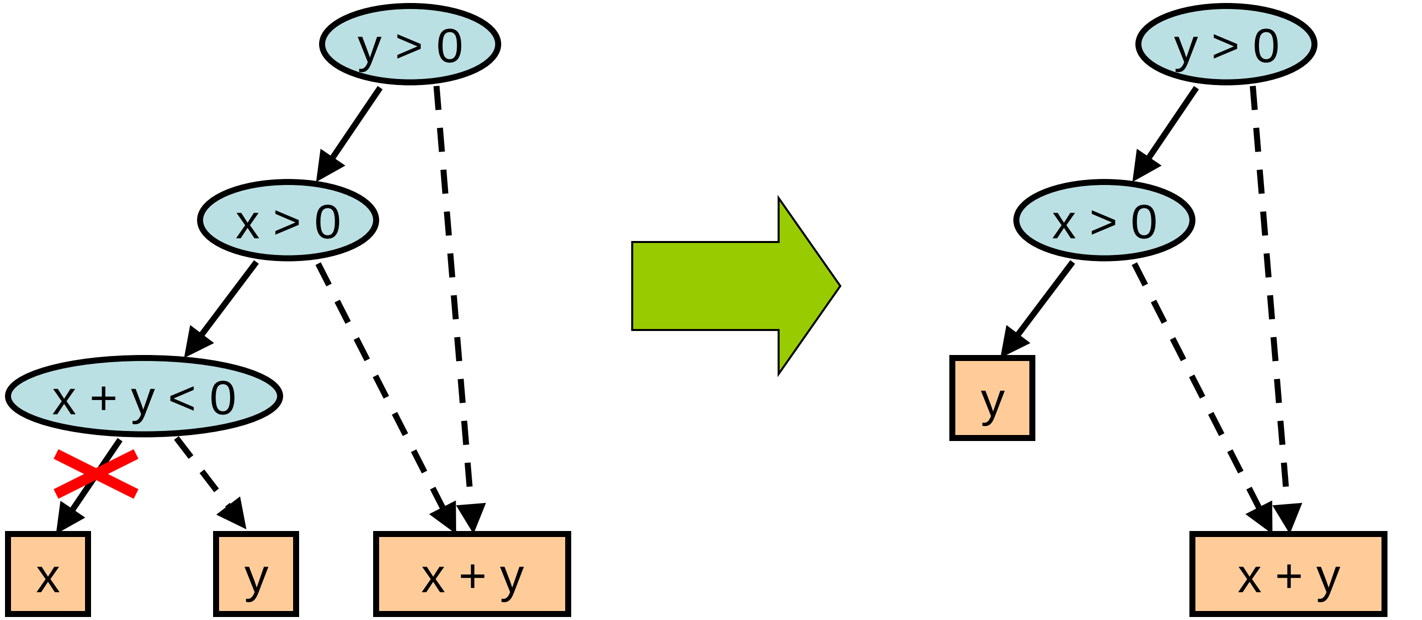


Correcting XADD Ordering

- Obtain *ordered* XADD from *unordered* XADD
  - key idea: binary operations maintain orderings



Maintaining Minimality



Node unreachable – x + y < 0 always false if x > 0 & y > 0

If **linear**, can detect with feasibility checker of LP solver & prune

More subtle prunings as well.

XADD Makes Possible all Previous Inference

Could not even do a single integral or maximization without it!

Applications

Expressive Closed-form Bayesian Inference

An Expressive Conjugate Prior for Bayesian Inference

- General Bayesian Inference

$$p(\vec{\theta}|D_{n+1}) \propto p(d_{n+1}|\vec{\theta})p(\vec{\theta}|D_n)$$

XADD XADD XADD

- Prior & likelihood for computational convenience?
  - No, choose as appropriate for your problem!

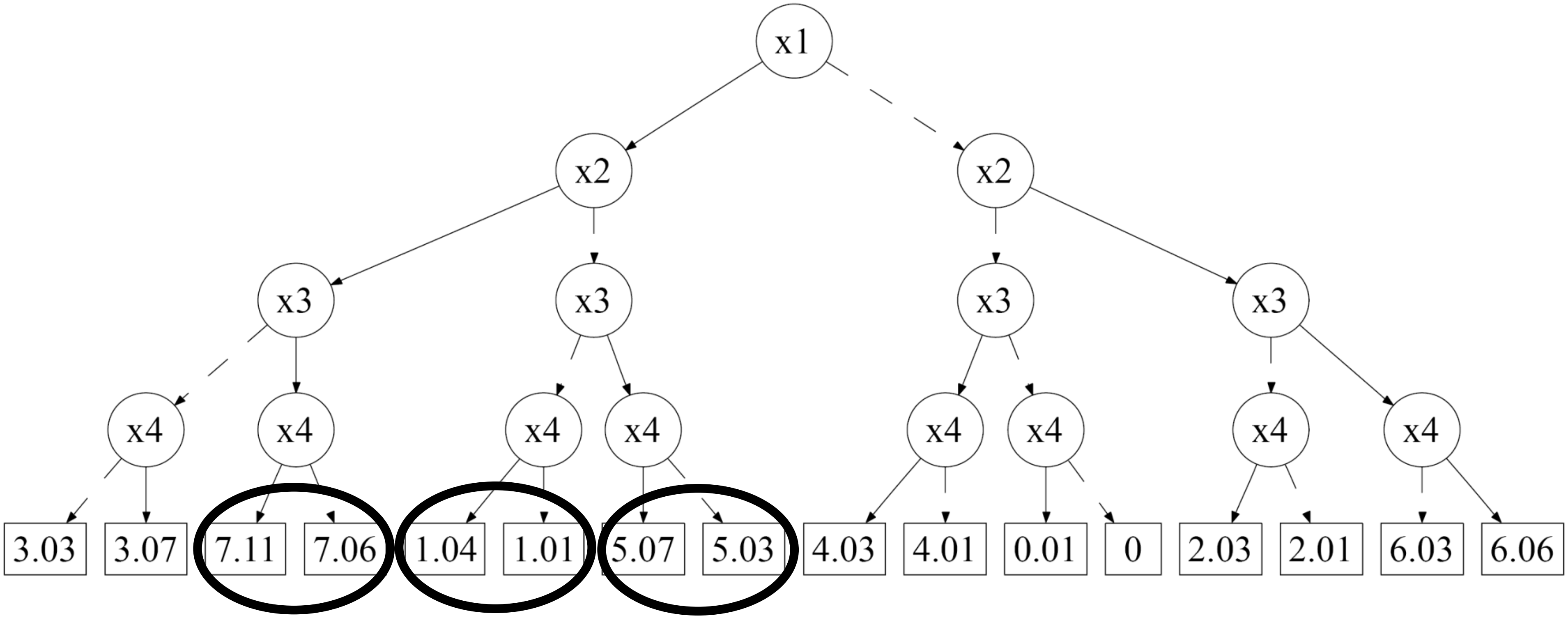
Approximate Inference

Sometimes no DD is compact,

but bounded approximation is…

Problem: Approximate an ADD?

- Sum: $(\sum_{i=1..3} 2^i \cdot x_i) + x_4 \cdot \varepsilon$-*Noise*

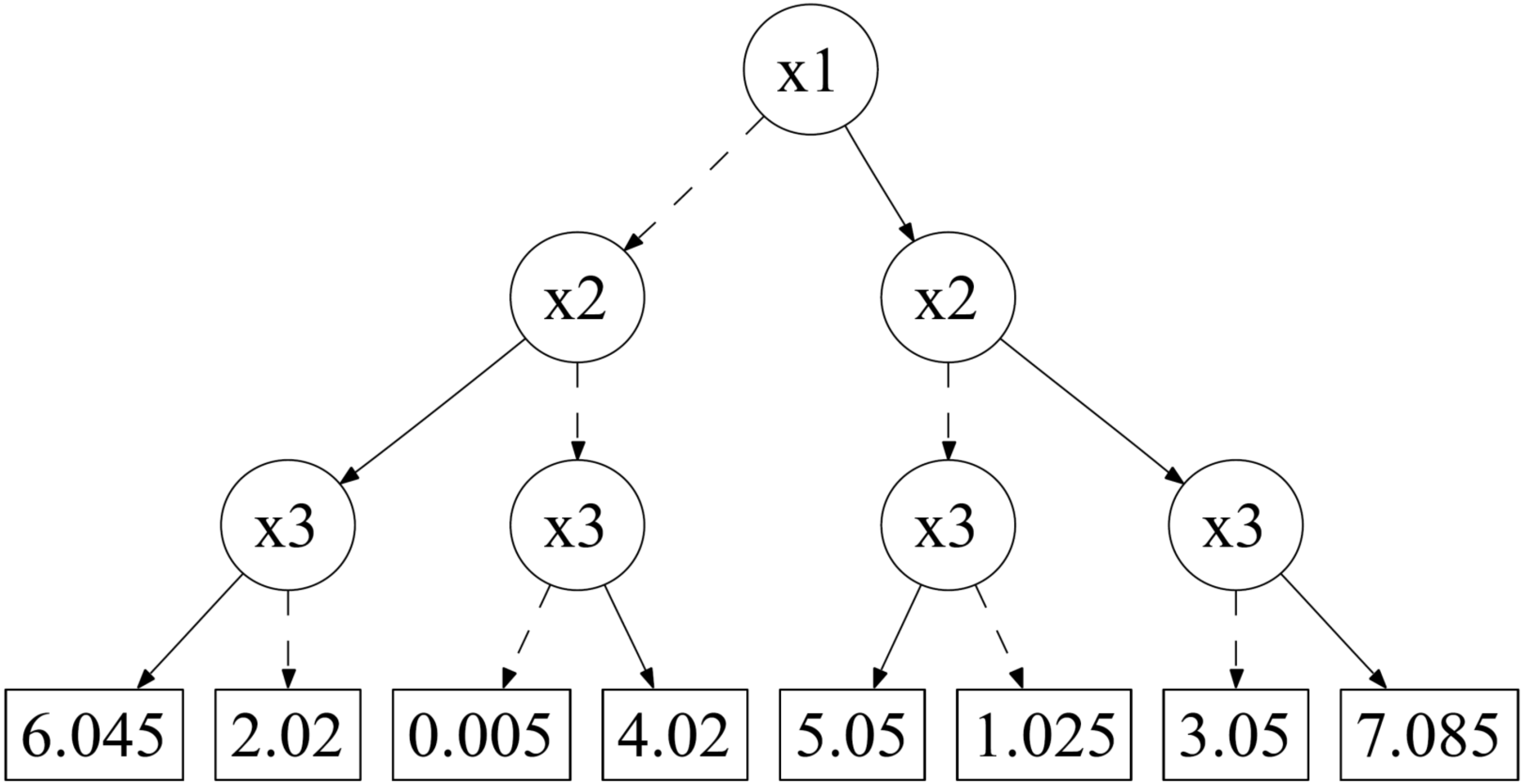


- How to approximate?

(St-Aubin, Hoey, Boutilier, NIPS-00)

Solution: APRICODD Trick

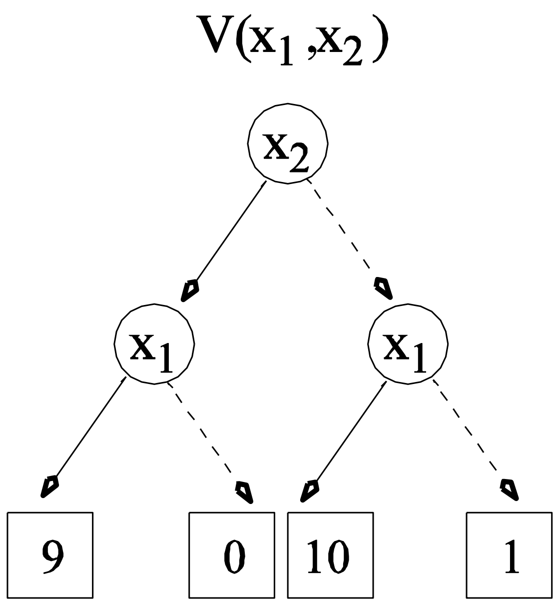
- Merge ≈ leaves and reduce:



- Error is bounded!

Can use ADD to Maintain Bounds!

- Change leaf to represent range [L, U]
  – Exact leaf is [V, V]
  – When merging leaves…
    - keep track of min and max values contributing

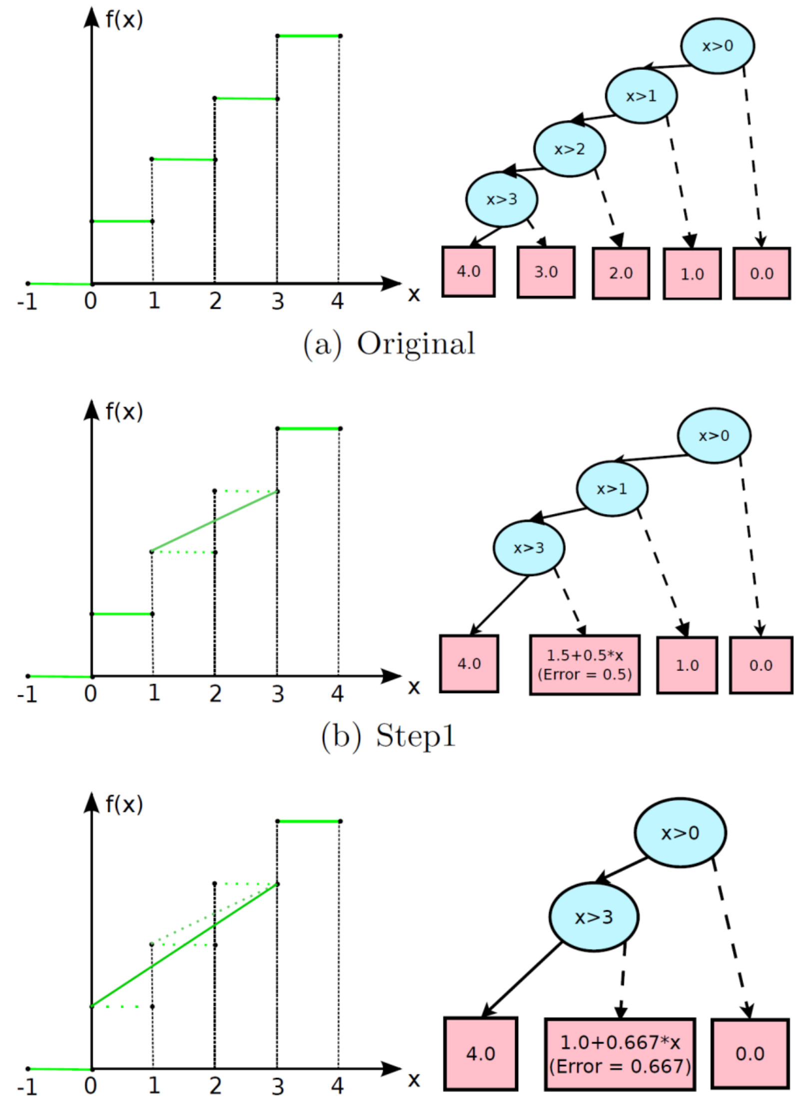


For operations, use “interval arithmetic”

XADD Approximation

- Can we extend APRICODD-style approximations to XADDs?

- Yes, but not as simple as averaging leaves…

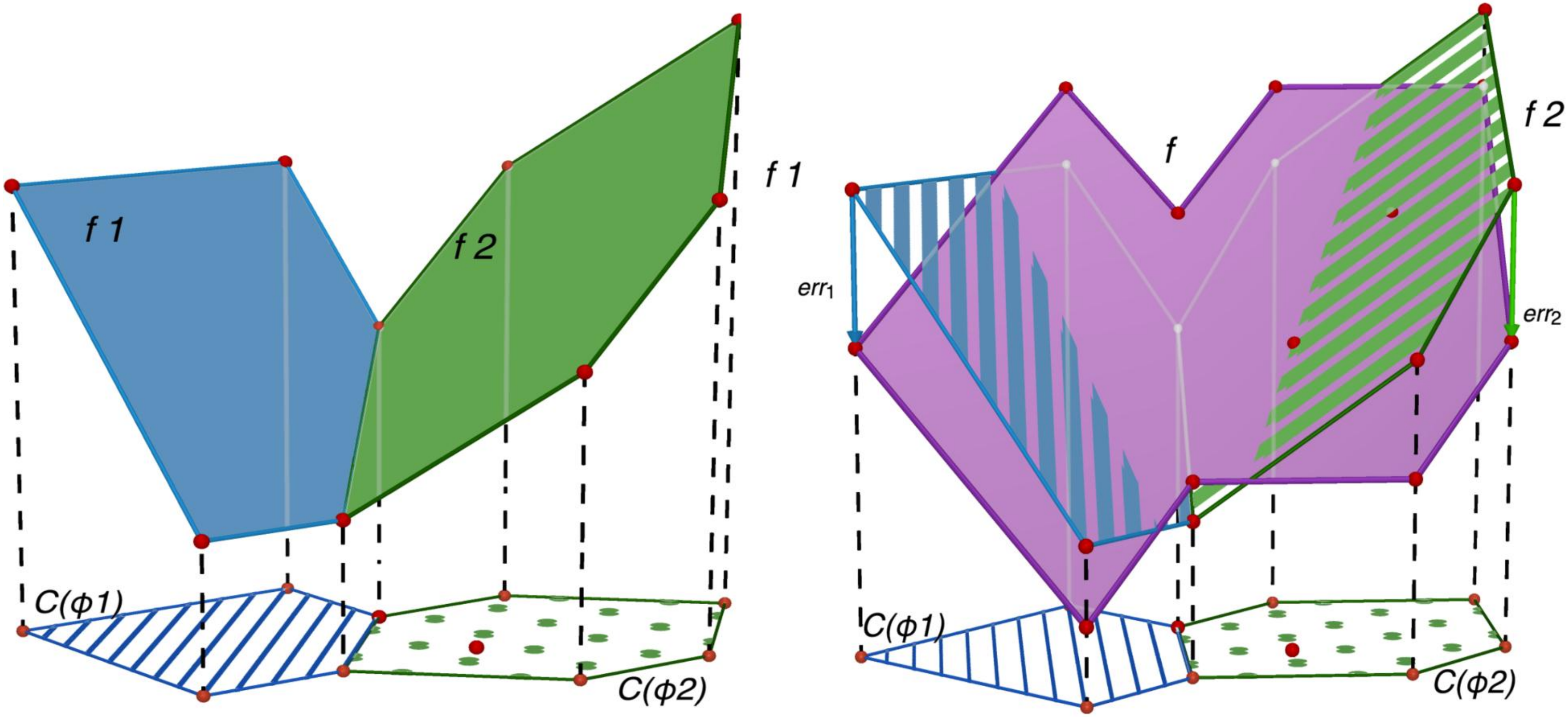


(a) Original

(b) Step1

(c) Step2

Linear XADD Leaf Merging

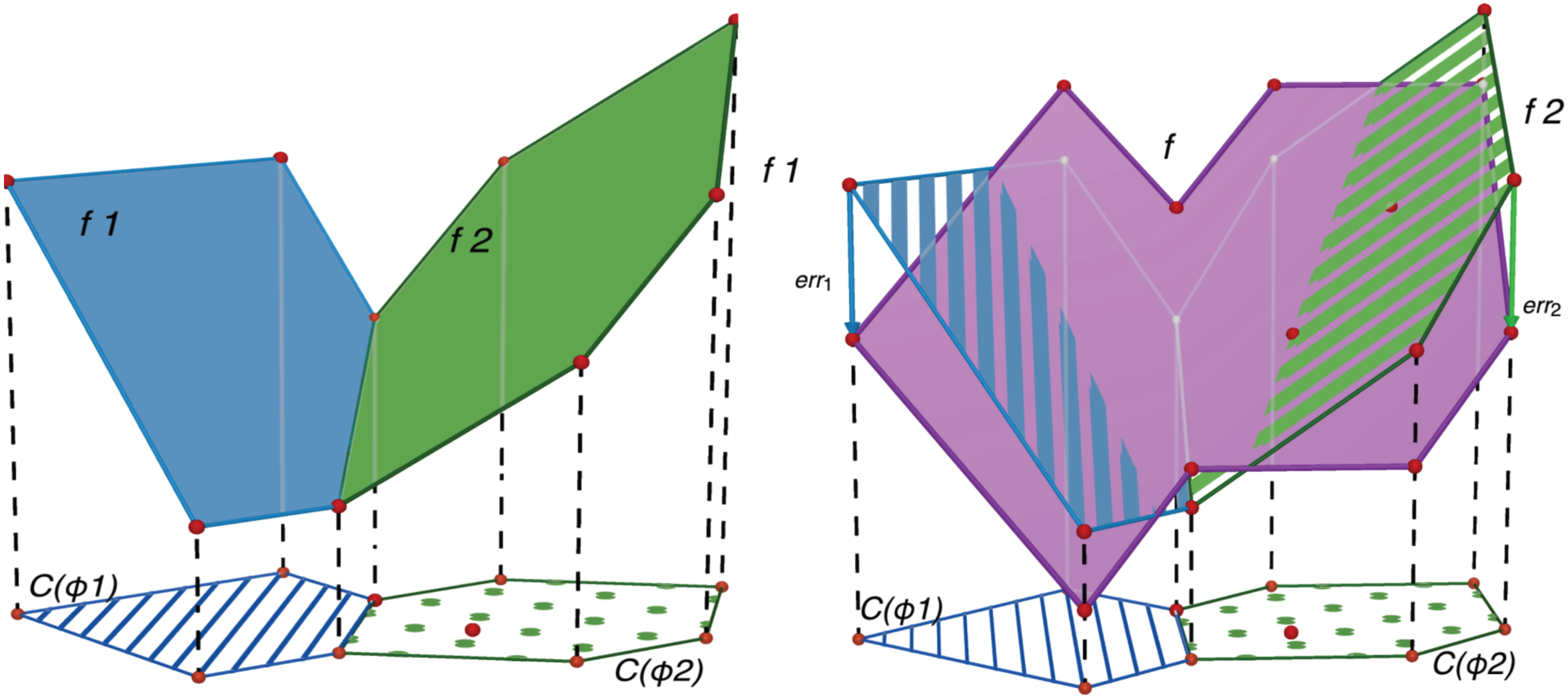


Best f*

Max error

$$\min_{\vec{c}^*} \boxed{\max_{i\in\{1,2\}} \max_{\vec{x}\in S_{\phi_i}} \left| \underbrace{\vec{c}_i^{\,T} \begin{bmatrix}\vec{x}\\1\end{bmatrix}}_{f_i} - \underbrace{\vec{c}^{*\,T}\begin{bmatrix}\vec{x}\\1\end{bmatrix}}_{f^*} \right|}$$

Linear XADD Leaf Merging

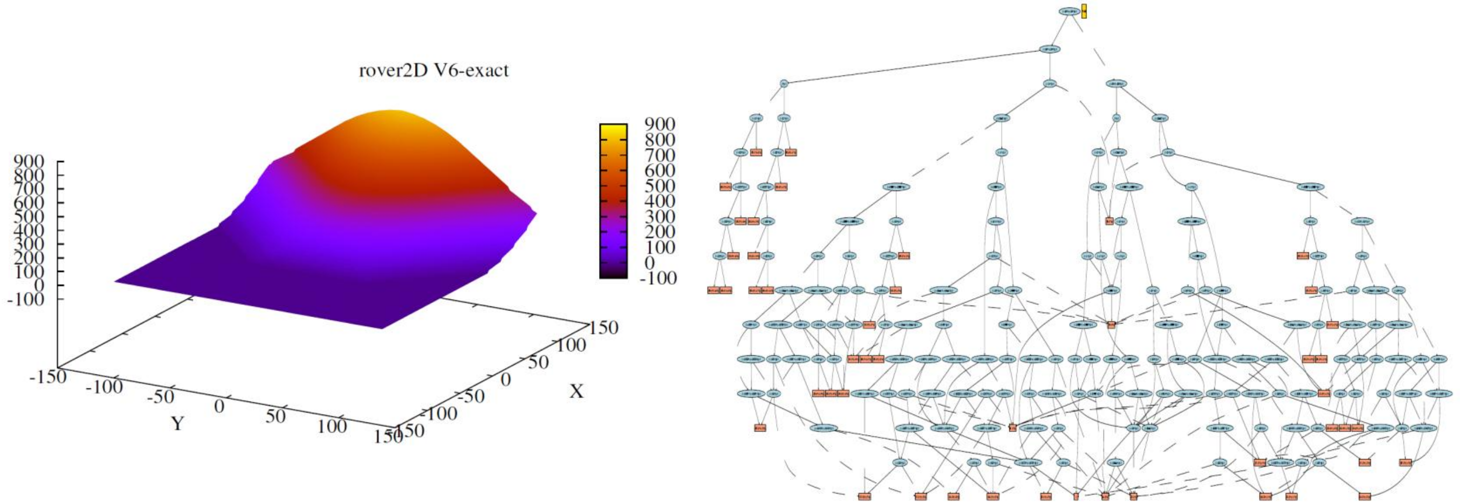


Constraint generation: for c*, use **LP** to generate max violated constraint

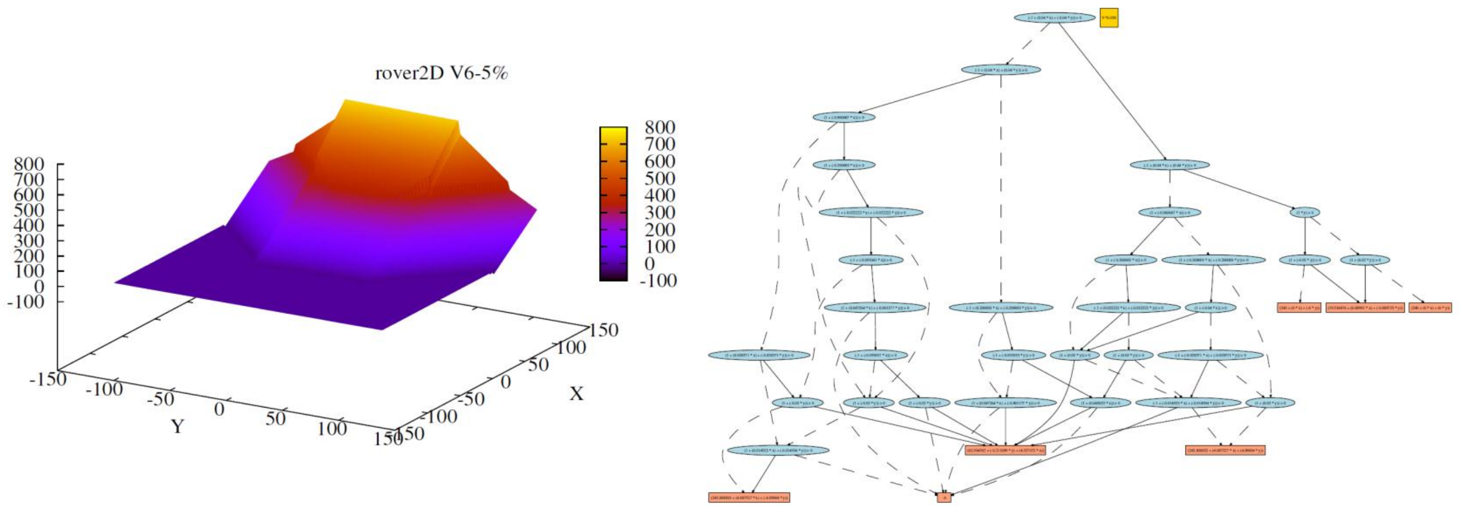
$$\min_{\vec{c}^*,\epsilon} \epsilon$$

$$s.t.\ \epsilon \geq \boxed{\left| \vec{c}_i^{\,T} \begin{bmatrix} \vec{x}_{ij}^k \\ 1 \end{bmatrix} - \vec{c}^{*\,T} \begin{bmatrix} \vec{x}_{ij}^k \\ 1 \end{bmatrix} \right| ; \quad \forall i \in \{1,2\}, \forall \theta_{ij}, \ \forall k \in \{1 \ldots N_{ij}\}}$$

Linear Approximation Example



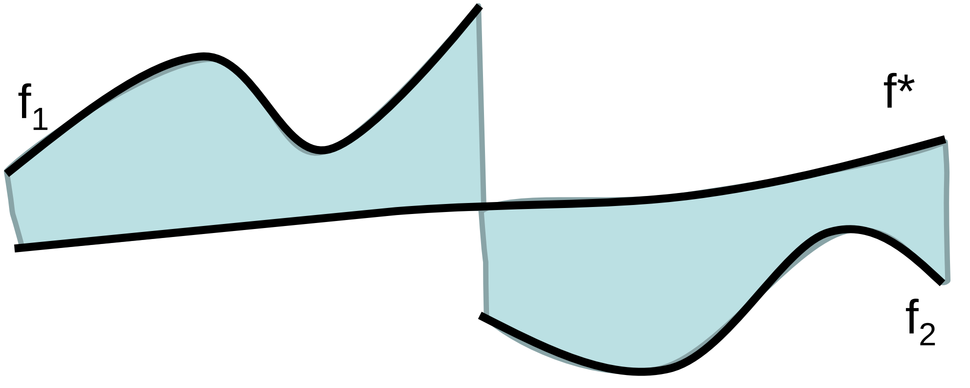
(a) Value at 6^{th} iteration for exact SDP.



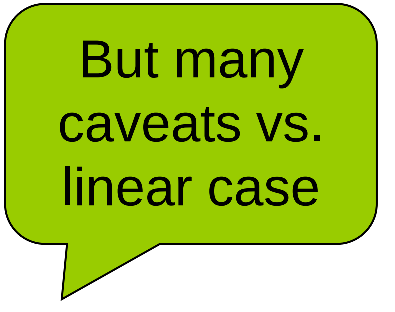
(b) Value at 6^{th} iteration for 5% approximate SDP.

Nonlinear XADD Approximation?

- 1D Example



- Questions
  - What approximating class?
  - What error function?
    - Max not feasible
    - Volume of squared error? Integral is exact.

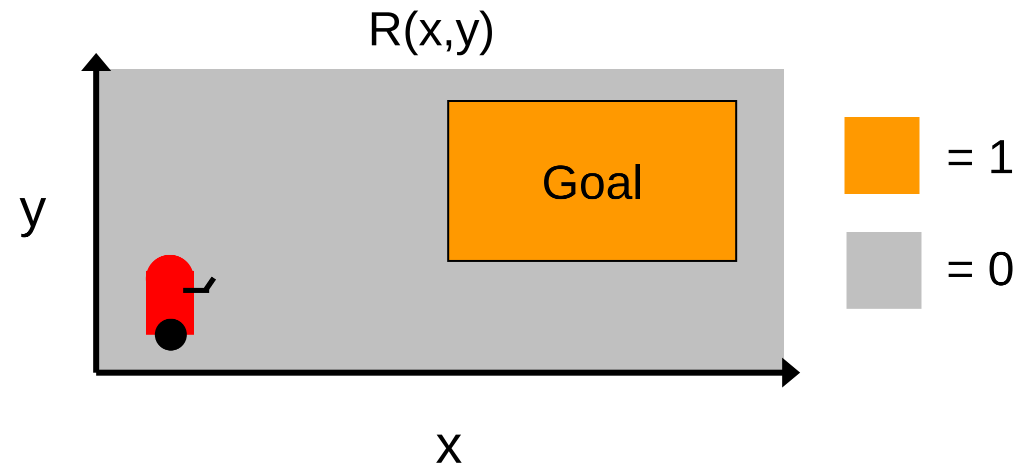


Applications

Optimal Sequential Decision-making

Continuous State MDPs

- 2-D Navigation
- State: $(x,y) \in \mathbb{R}^2$
- Actions:
  - move-x-2
    - x’ = x + 2
    - y’ = y
  - move-y-2
    - x’ = x
    - y’ = y + 2
- Reward:
  - R(x,y) = I[(x > 5) ^ (x < 10) ^ (y > 2) ^ (y < 5)]

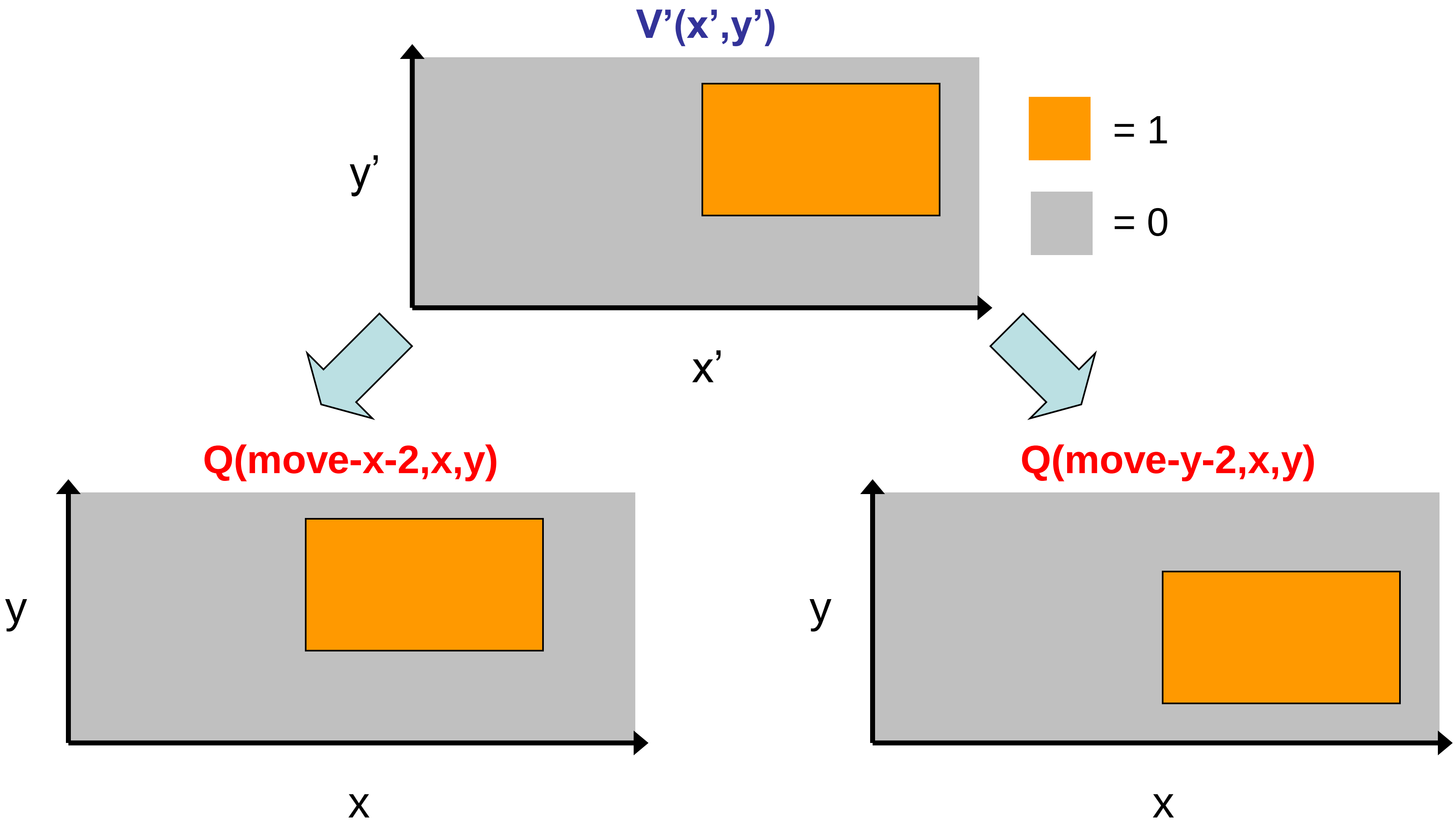


Feng *et al* (UAI-04) Assumptions:

1. Continuous transitions are deterministic and linear
2. Discrete transitions can be stochastic
3. Reward is piecewise rectilinear convex

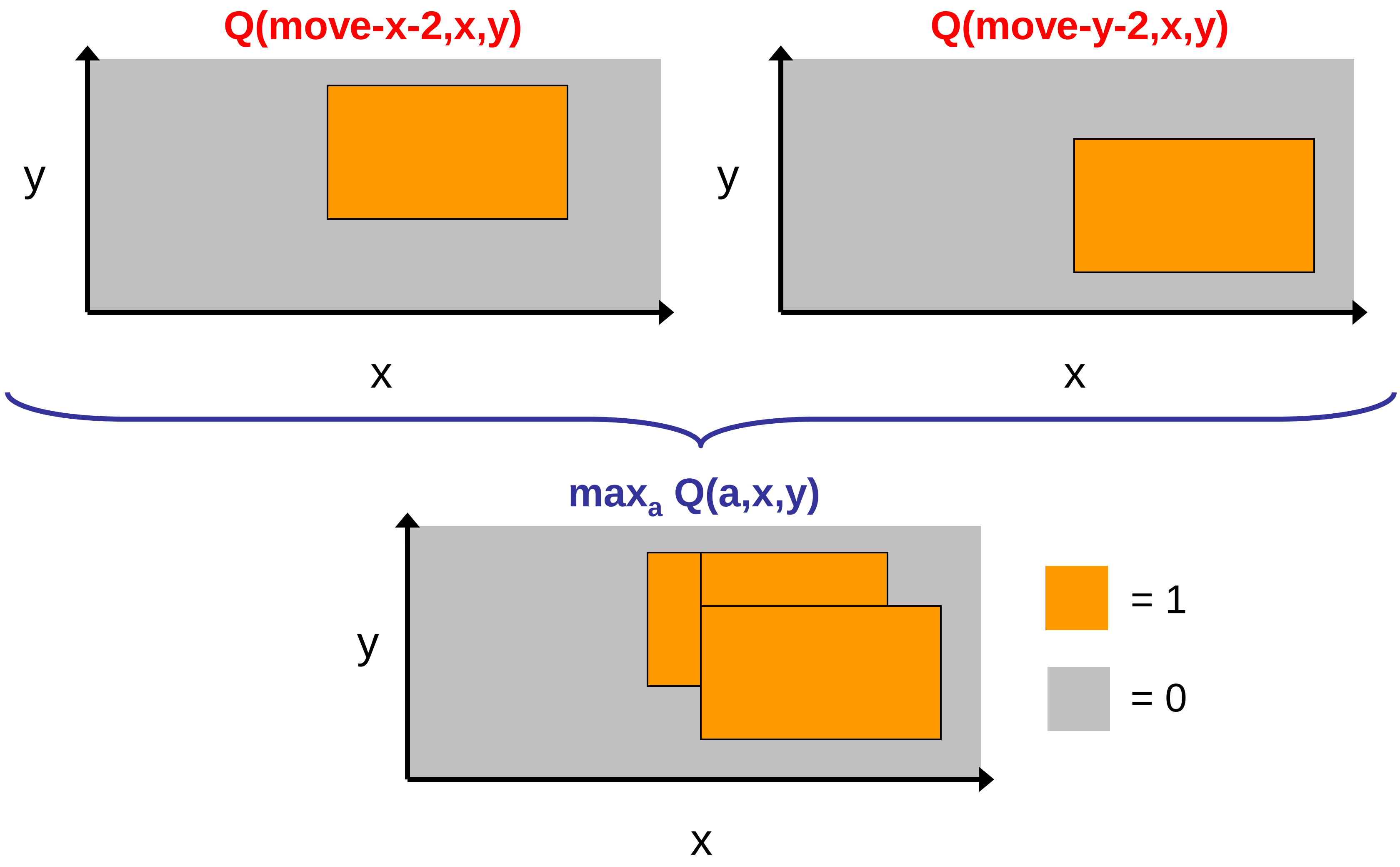
Exact Solutions to DC-MDPs: Regression

- Continuous regression is just translation of “pieces”



Exact Solutions to DC-MDPs: Maximization

- Q-value maximization yields piecewise rectilinear solution

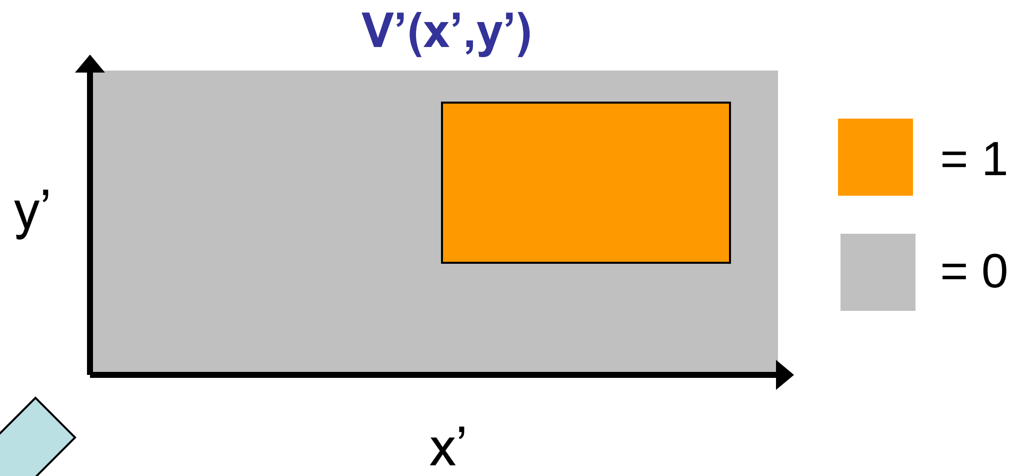


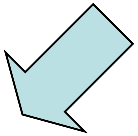
Previous Work Limitations I

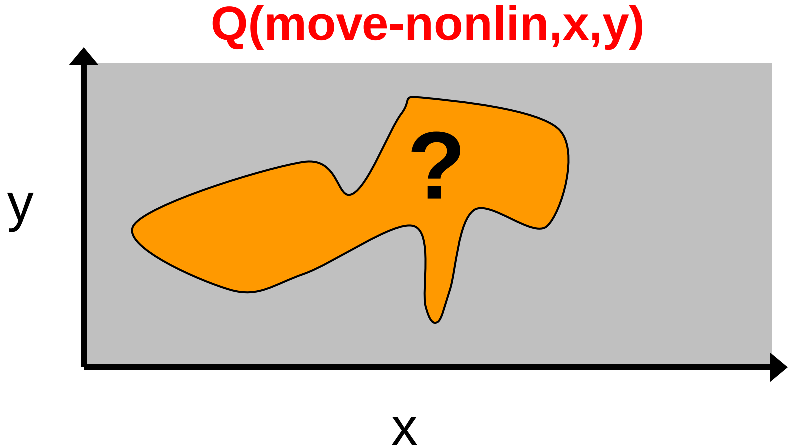
- Exact regression when transitions nonlinear?

Action move-nonlin:

- $x' = x^3 y + y^2$
- $y' = y * \log(x^2 y)$



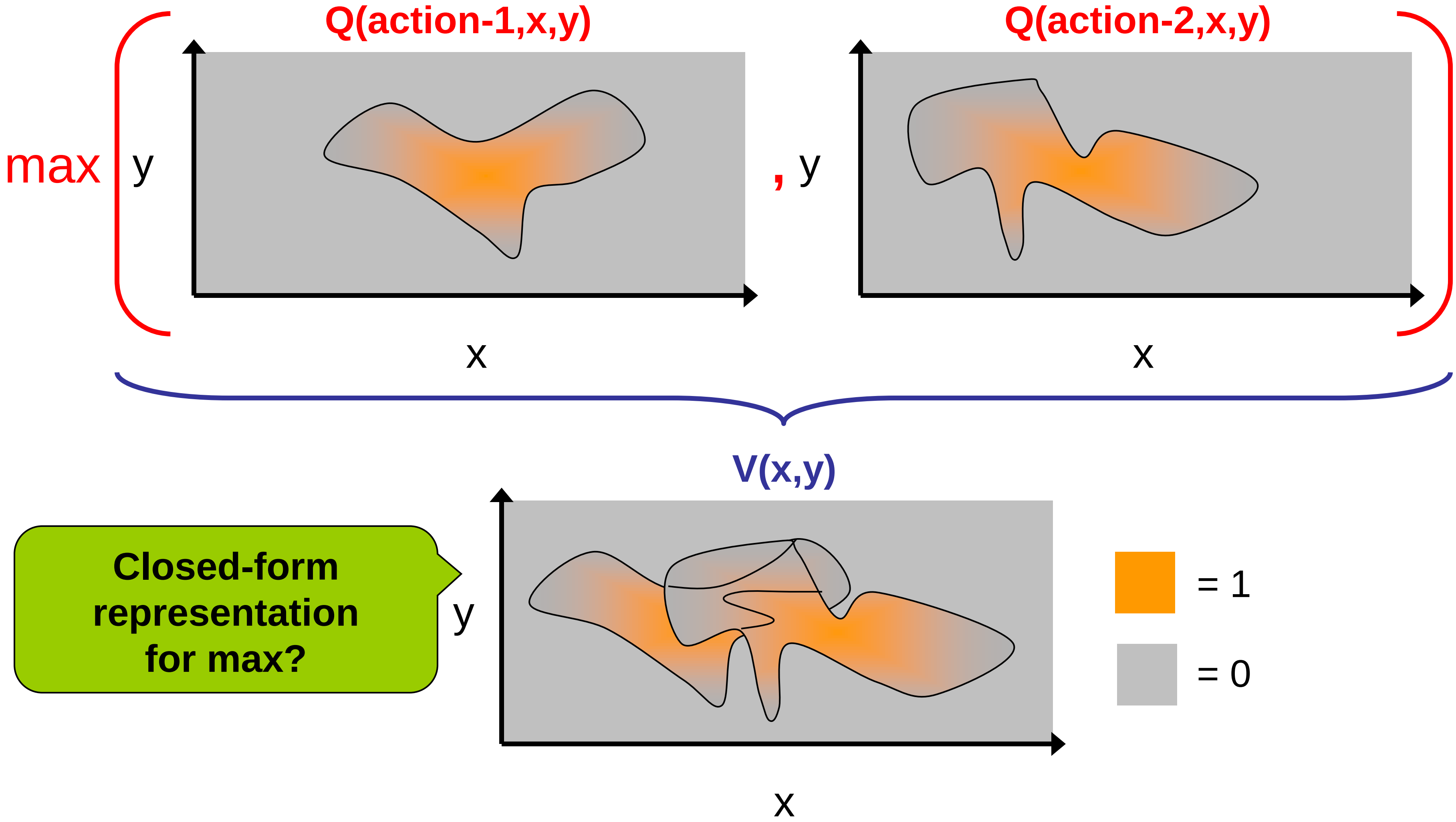




How to compute boundary in closed-form?

Previous Work Limitations II

- max(.,.) when reward/value arbitrary piecewise?



A solution to previous limitations:

Symbolic Dynamic Programming (SDP)

n.b., motivated by SDP from Boutilier *et al* (IJCAI-01) but here continuous instead of relational

Symbolic Dynamic Programming

- Value Iteration for h∈0..H
  - Regression step:

$$Q_a^{h+1}(\vec{b},\vec{x}) = R_a(\vec{b},\vec{x}) + \gamma \cdot$$

$$\sum_{\vec{b}'} \int_{\vec{x}'} \left(\prod_{i=1}^{n} P(b_i'|\vec{b},\vec{x},a) \prod_{j=1}^{m} P(x_j'|\vec{b},\vec{b}',\vec{x},a) \right) V^h(\vec{b}',\vec{x}')d\vec{x}'$$

Due to deterministic assumption, these are δ functions

  - Maximization step:

$$V_{h+1}(\vec{b},\vec{x}) = \max_{a \in A} Q_a^{h+1}(\vec{b},\vec{x})$$

SDP Regression Step

- **Continuous variables** x_j
  - $\int_x \delta[x - y] f(x) dx = f(y)$ triggers *symbolic substitution*
  - More complex: $\int_{x'_j} \delta[x'_j - g(\vec{x})] V' dx'_j \; = \; V'\{x'_j / g(\vec{x})\}$

$$\int_{x'_1} \delta[x'_1 - (x_1^2 + 1)] \left(\begin{cases} \underline{x'_1} < 2 : & \underline{x'_1} \\ \underline{x'_1} \geq 2 : & \underline{x'^2_1} \end{cases} \right) dx'_1 \; = \; \begin{cases} \underline{x_1^2 + 1 < 2} : & \underline{x_1^2 + 1} \\ \underline{x_1^2 + 1 \geq 2} : & \underline{(x_1^2 + 1)^2} \end{cases}$$

  - General case: *g* is *case* – need *conditional substitution*
    - See UAI-11 paper

SDP for Continuous State MDPs

Symbolic Dynamic Programming (SDP)… exact closed-form solution for **any** continuous state MDP!

- Value Iteration for h∈0..H
  - Regression step:

XADD

$$Q_a^{h+1}(\vec{b},\vec{x}) = R_a(\vec{b},\vec{x}) + \gamma \cdot$$

XADD XADD XADD

$$\sum_{\vec{b}'} \int_{\vec{x}'} \left(\prod_{i=1}^{n} P(b_i'|\vec{b},\vec{x},a) \prod_{j=1}^{m} P(x_j'|\vec{b},\vec{b}',\vec{x},a) \right) V^h(\vec{b}',\vec{x}') d\vec{x}'$$

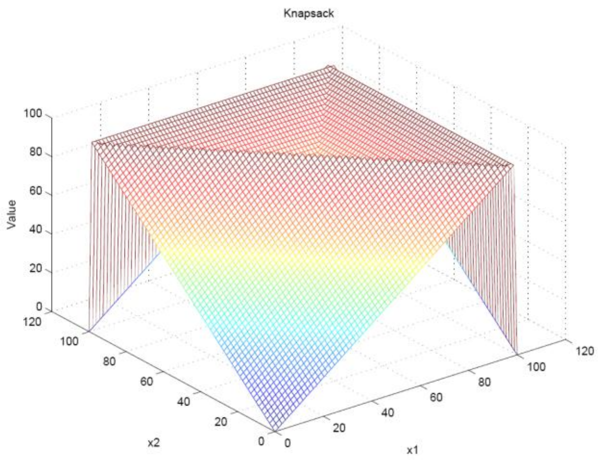
  - Maximization step:

$$V_{h+1} = \max_{a \in A} Q_a^{h+1}(\vec{b},\vec{x})$$

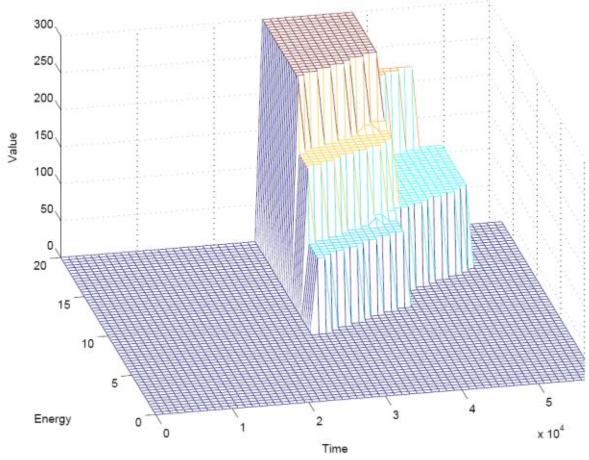
XADD XADD

That's SDP! 3D Value Function Gallery

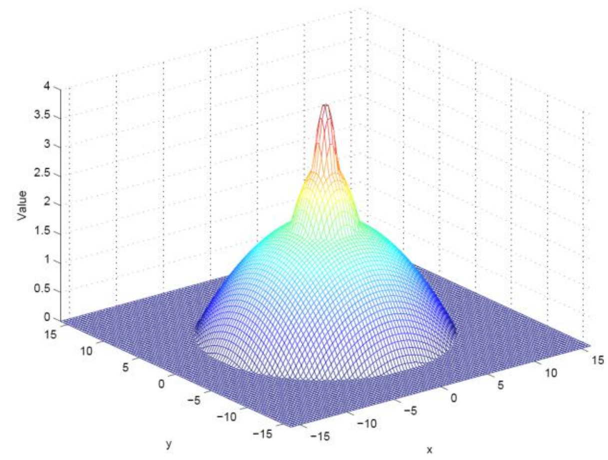
Knapsack



Mars Rover Linear



Mars Rover Nonlinear



Exact value functions in case form:

- Arbitrary transitions, reward **(not just polynomial)**
- (non)linear piece boundaries and function surfaces!

Continuous Actions?

If we can solve this, can solve **multivariate inventory control** – closed-form policy unknown for 50+ years!

Continuous Actions

- Inventory control
  - Reorder based on stock, future demand
  - Action: $a(\vec{\Delta}); \vec{\Delta} \in \mathbb{R}^{|a|}$

- Need max$_{\Delta}$ in Bellman backup

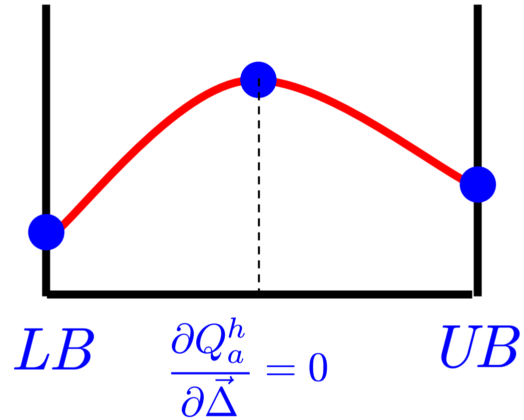
$$V_{h+1} = \max_{a \in A} \max_{\vec{\Delta}} Q_a^{h+1}(\vec{b}, \vec{x}, \vec{\Delta})$$

- How to do max$_{\Delta}$?
  - And track maximizing Δ substitutions to recover π?

Max-out Case Operation

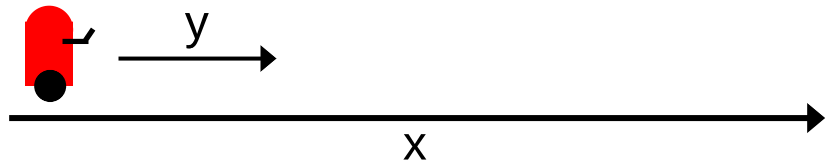
- max$_x$ case(x) like $\int_x$ case(x), can reduce to single partition max
  - In a *single* case partition …*max* w.r.t. critical points
    - LB, UB
    - Derivative is zero (Der0)

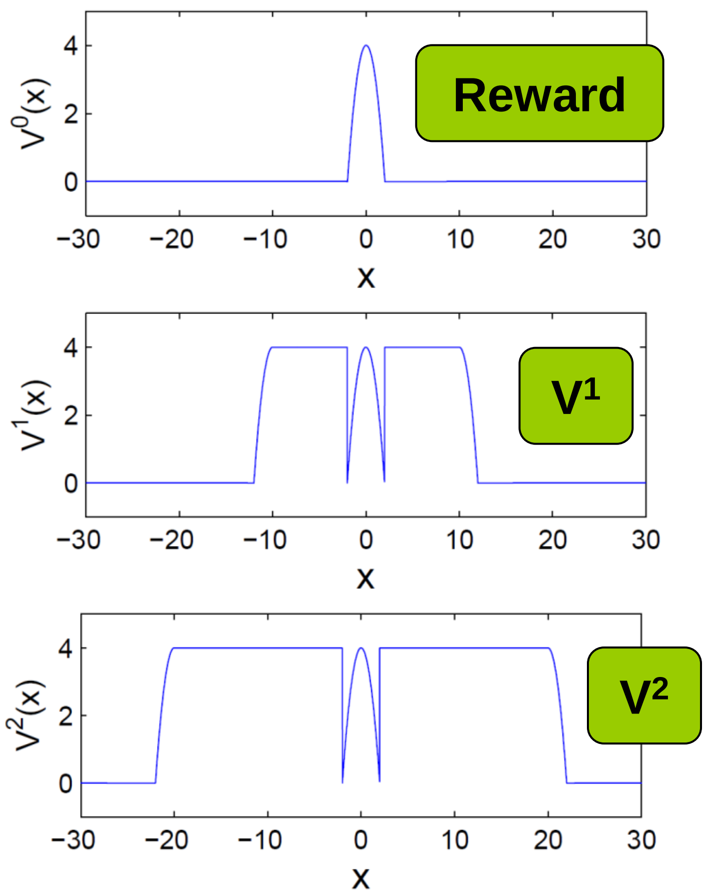
    - max(case(x/LB),
      case(x/UB),
      case(x/Der0))
  - Can even track substitutions through max to recover function of maximizing assignments!

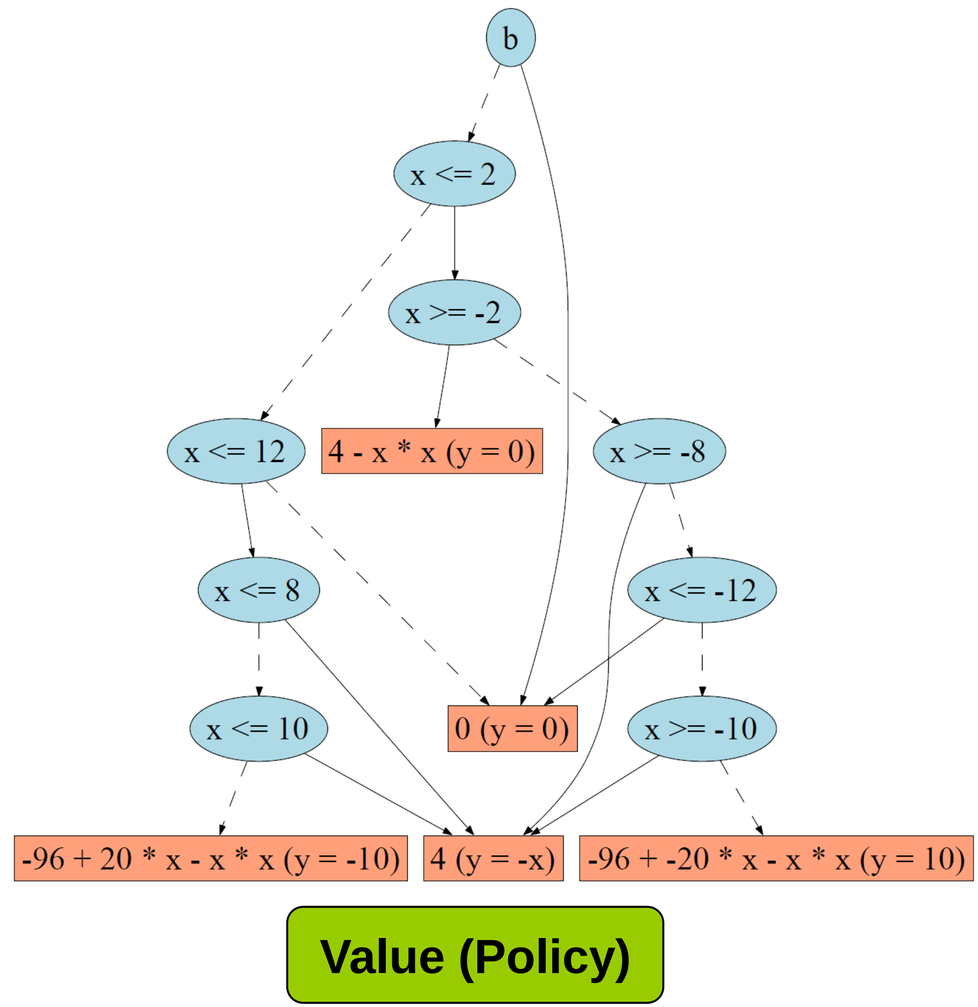


See UAI 2011, AAAI 2012 papers for more details

Illustrative Value and Policy



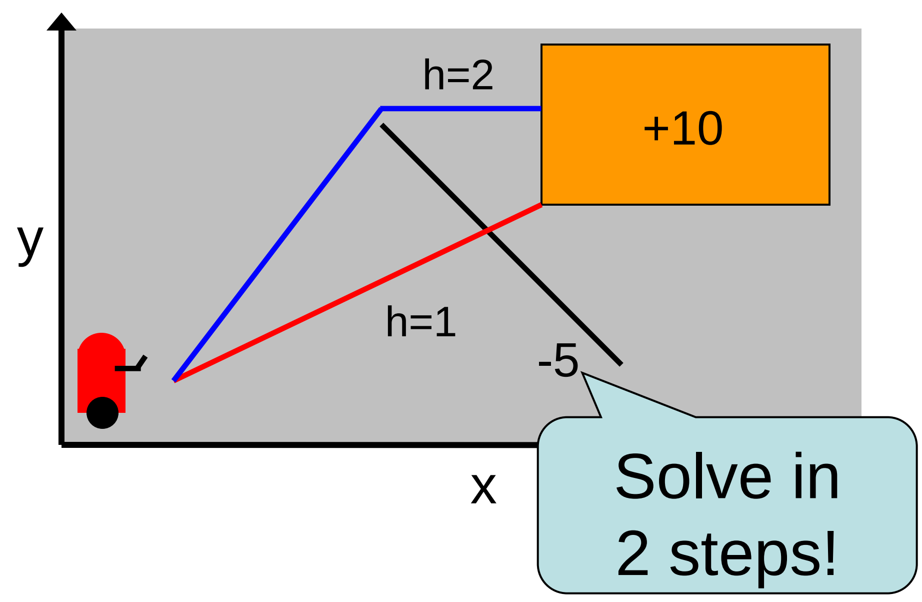




Continuous Actions, Nonlinear

- **Robotics**
  - Continuous position, joint angles
  - Represent exactly with polynomials
    - Radius constraints

- **Obstacle Navigation**
  - 2D, 3D, 4D (time)
  - Don’t discretize!
    - ~~Grid worlds~~
  - But nonlinear ☹



Multilinear, quadratic extensions.

In general: algebraic geometry.

Sequential Control Summary

- Continuous state, action, observation (PO)MDPs
  - Discrete action MDPs **UAI-11**
  - Continuous action MDPs (incl. exact policy) **AAAI-12b**
  - Continuous observation POMDPs **NIPS-12**
  - Robust solutions with continuous noise **IJCAI-13**
  - Multilinear, quadratic extensions **In progress**

Applications

Constrained Optimization

$\max_x \text{case}(x)$ = Constrained Optimization!

- Conditional constraints
  - E.g., if $(x > y)$ then $(y < z)$
  - MILP, MIQP equivalent
- Factored / sparse constraints
  - Constraints may be sparse!

    $x_1 > x_2, \quad x_2 > x_3, \quad \ldots, \quad x_{n-1} > x_n$
  - Dynamic programming for continuous optimization!
- Parameterized optimization
  - $f(y) = \max_x f(x,y)$
  - Maximum value, substitution as a **function of y**

Recap

- Defined a calculus for piecewise functions
    - $f_1 \oplus f_2$, $f_1 \otimes f_2$
    - $\max(f_1, f_2)$, $\min(f_1, f_2)$
    - $\int_x f(x)$
    - $\max_x f(x)$, $\min_x f(x)$

- **Defined XADD to efficiently compute with cases**

- **Makes possible**
    - **Exact inference in continuous graphical models**
    - **First exact solutions to planning, control, and OR problems**
    - **New paradigms for optimization**

Symbolic Piecewise Calculus + XADD

= Expressive Continuous Inference, Optimization, & Control

Thank you!

Questions?