PCAA

Polymorphous Cognitive Agent Architecture

Enabling Cognitive Architectures for UAV Mission Planning

- Mohammed Amduka (CO-PI), Lockheed Martin ATL, mamduka@atl.lmco.com,
- Boris Gelfand (CO-PI), Lockheed Martin ATL, belfand@atl.lmco.com,
- Jon C. Russo, Lockheed Martin ATL, jrusso@atl.lmco.com,
- **Keith Pedersen, Lockheed Martin Aero, keith.pedersen@lmco.com,
- **Richard Lethin, Reservoir Labs, lethin@reservoir.com,
- Jonathan Springer, Reservoir Labs, springer@reservoir.com,
- Rajit Manohar. Cornell University, rajit@cs.cornell.edu,
- Rami Melhem. University of Pittsburgh, melhem@cs.pitt.edu

***presenters*

Sponsored by DARPA/IPTO in the ACIP program; Dr. William J. Harrod, PM

FA8750-04-C-0266

PCAA Team

LOCKHEED MARTIN
Advanced Technology Laboratories
ATL

Brian Boesch: Technology Director
Mohammed Amduka: Co-PI
Boris Gelfand: Co-PI

- **Technology Architecture and Integration**
- **Project Management**

ATL *Dan Davenport* *Group Lead*

ATL *Krishna Jha* *Group Lead*

ATL *Jon Russo* *Group Lead*

Applications

LOCKHEED MARTIN
Aeronautics Advanced Development Programs
ADP

LOCKHEED MARTIN
Integrated Systems and Solutions

Cognitive Architecture

reservoir labs

Soar Technology
Thinking *inside* the box.

NewVectors LLC
An Altarum Subsidiary

Micro Analysis & Design

Carnegie Mellon University

Computation Medium Layer

CALTECH

University of Pittsburgh

Cornell University

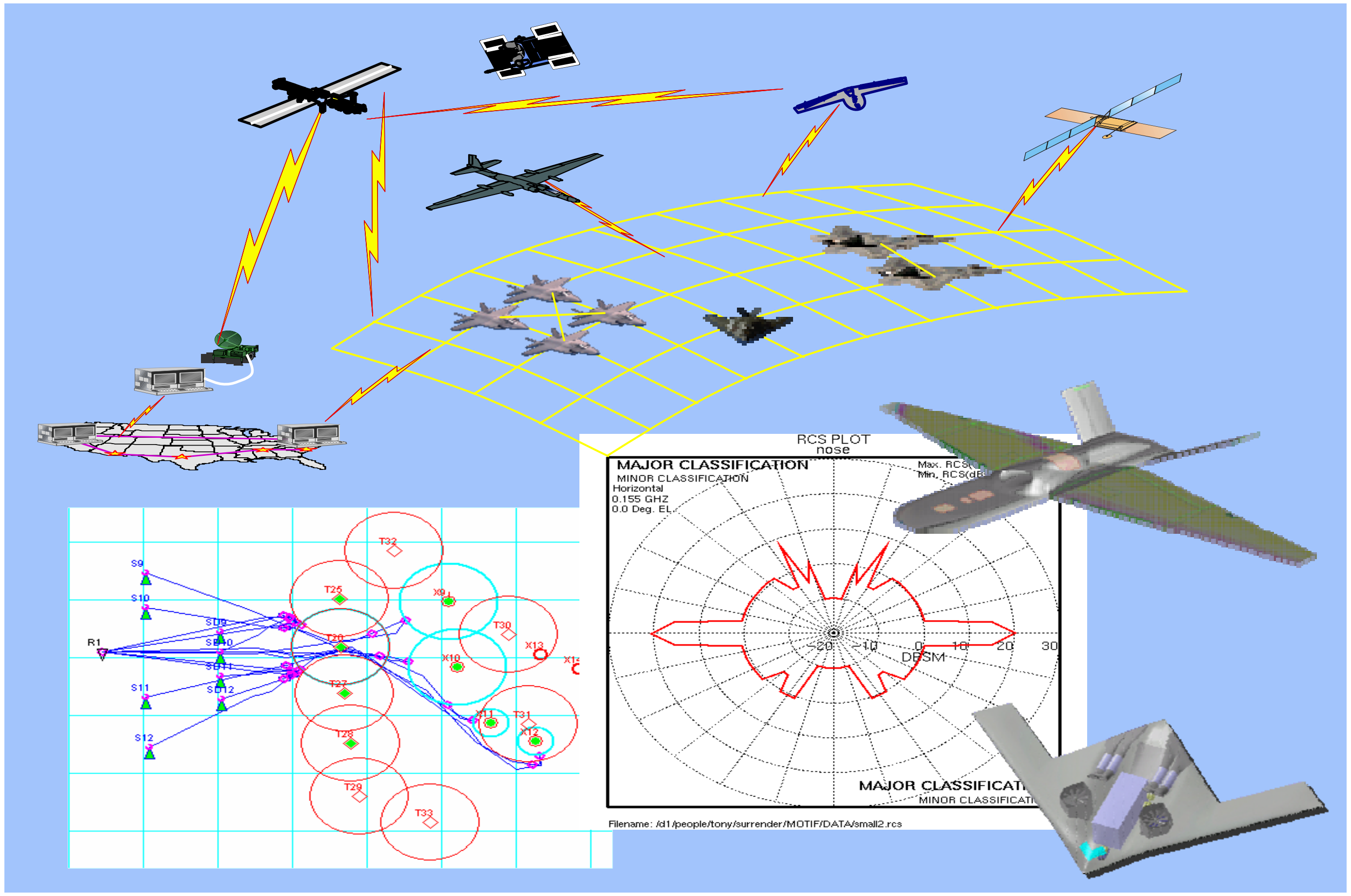
University of Texas at Austin

Presentation Outline

- Autonomous vehicles demand new approaches for algorithm design and computation
- We are experimenting with “cognitive techniques” to address limitations of “algorithmic techniques”
- We are developing new system and hardware elements to improve the performance and applicability of the “cognitive techniques”

Dynamic Multi-UAV Mission Planning

UAV Search and Destroy (SAD) Mission Planning



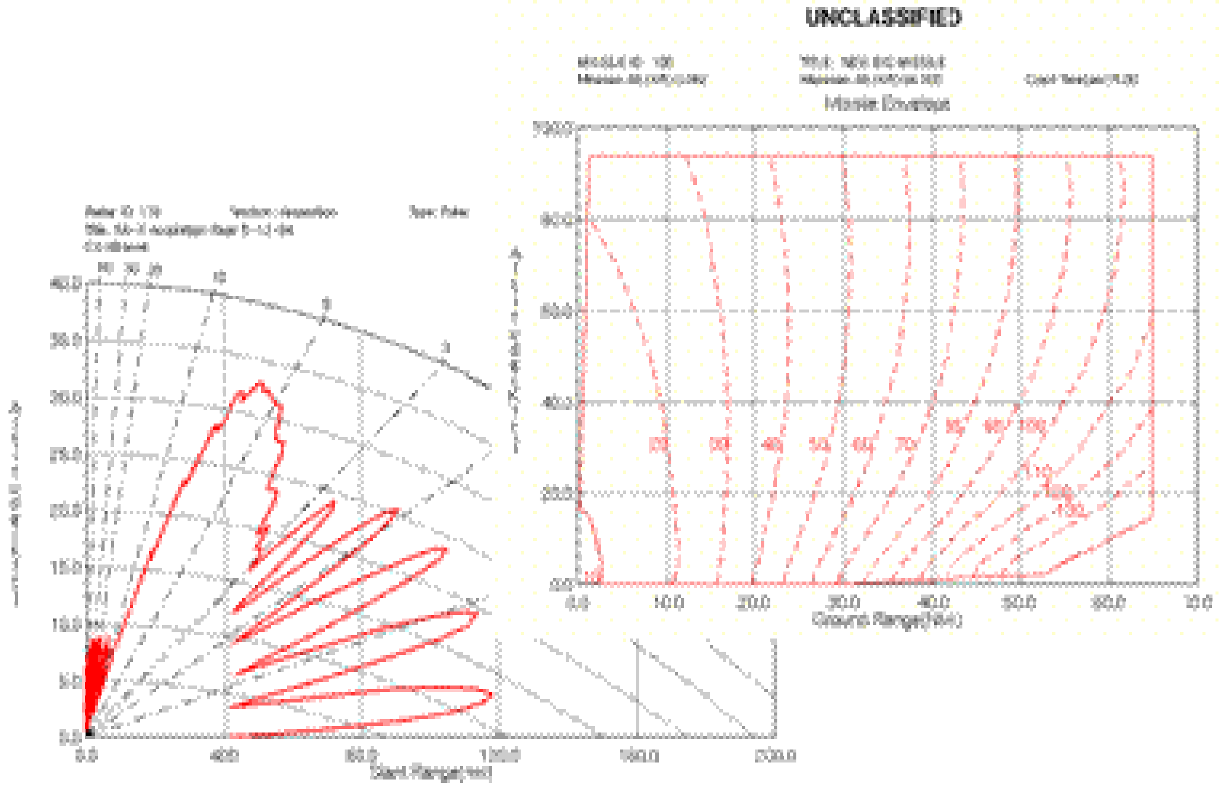
Fielded Mission Planning Systems

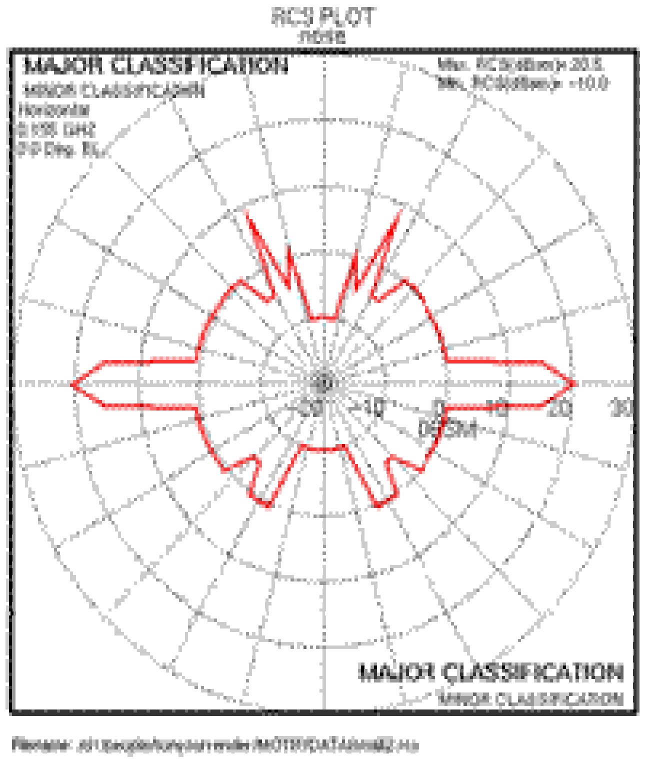
F-117

F-22

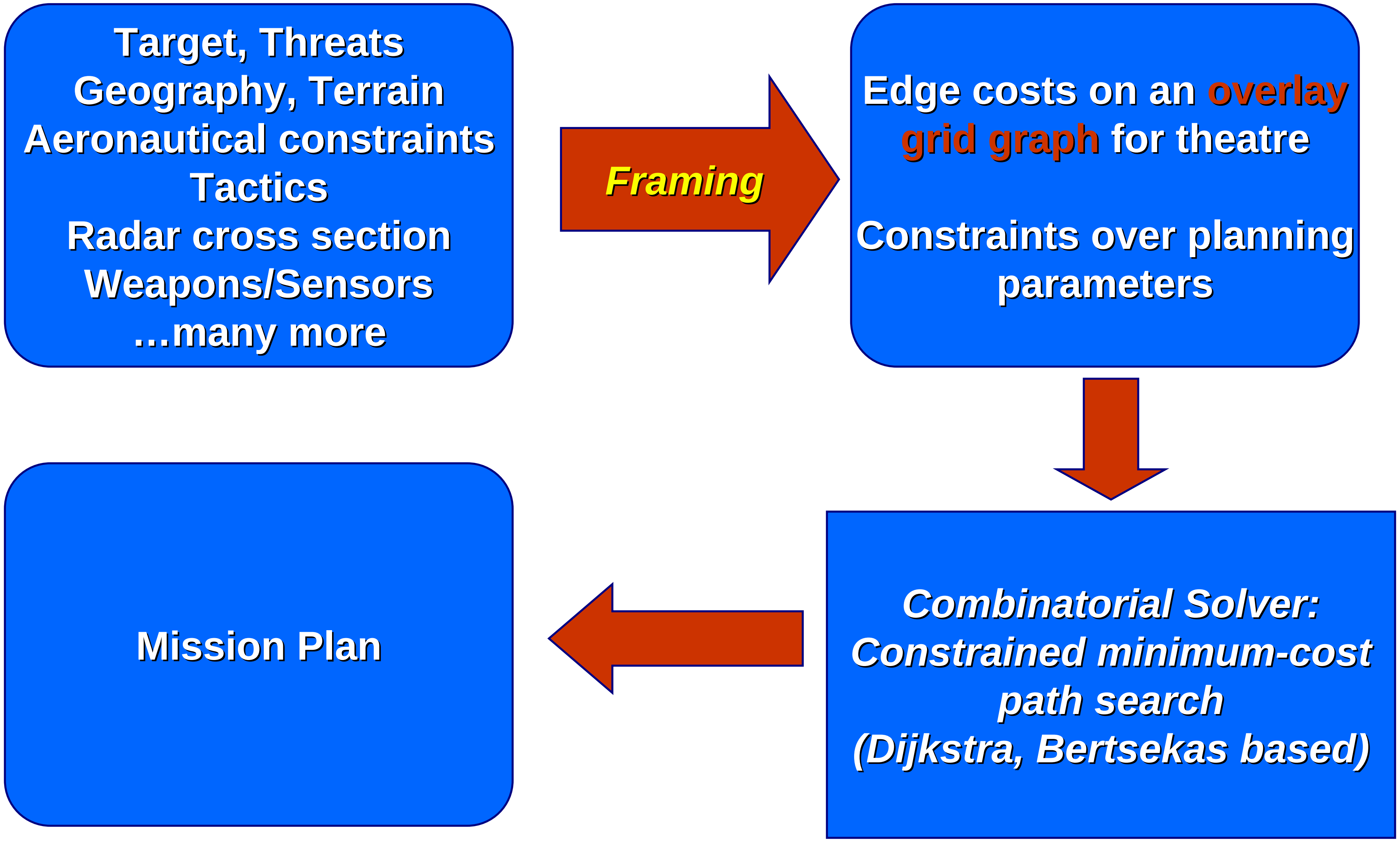
- **Fielded Systems Manage:**
  - Targets, threats
  - Geographical, terrain
  - Aeronautical constraints
  - Tactics
  - Radar cross section
  - Weapons/Sensors
  - …much more

“Fuzzball”


UNCLASSIFIED
Missile Envelope
Ground Range(NM)
Slant Range(nm)


RCS PLOT
MAJOR CLASSIFICATION
MINOR CLASSIFICATION
Horizontal
dBSM
MAJOR CLASSIFICATION
MINOR CLASSIFICATION

Fielded Systems: Algorithmic Approach


Target, Threats
Geography, Terrain
Aeronautical constraints
Tactics
Radar cross section
Weapons/Sensors
…many more
Framing
Edge costs on an overlay grid graph for theatre
Constraints over planning parameters
Combinatorial Solver: Constrained minimum-cost path search (Dijkstra, Bertsekas based)
Mission Plan

Processing Benchmarks

"Toy" Scenario

600 x 800 nm
2nd Neighbor
5nm Edge
4 Alt (7,10,15,20K)
3 Mission Objectives
1.7 GHz Processor
Constrained
~ 30 seconds

Current fielded systems can require <u>minutes or hours</u> of computation time on large cluster

Thus largely held to pre-mission planning

Dynamic Multi-UAV MP

- **Multiple UAVs**
  - Collaboration
  - Assignment
  - Routing
- **Larger regions, finer granularities**
- **More complicated constraints**
  - Route deconfliction
  - Complex tactics
- **Contingencies**
  - New targets/threats
  - Weather

Exponential Increase in Problem Complexity

Dynamic Requirement

Making “algorithmic technique” tractable

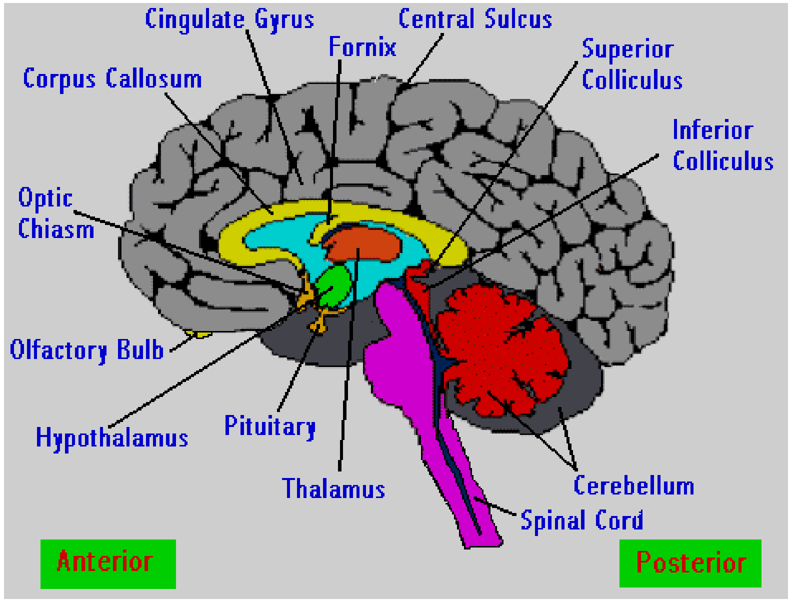
- Limit dimensions (grid size, LO fuzzball)
- Avoid symbolic constructs Employ suboptimal heuristics Abstract to use highly linear solvers
- Separate parameters and approximate in phases
- Refine from previously computed plan
- Labor, labor, labor …

But faced with exponential problem complexity growth, is there a better way?

Cognitive Approach

Inspirations from an Architecture

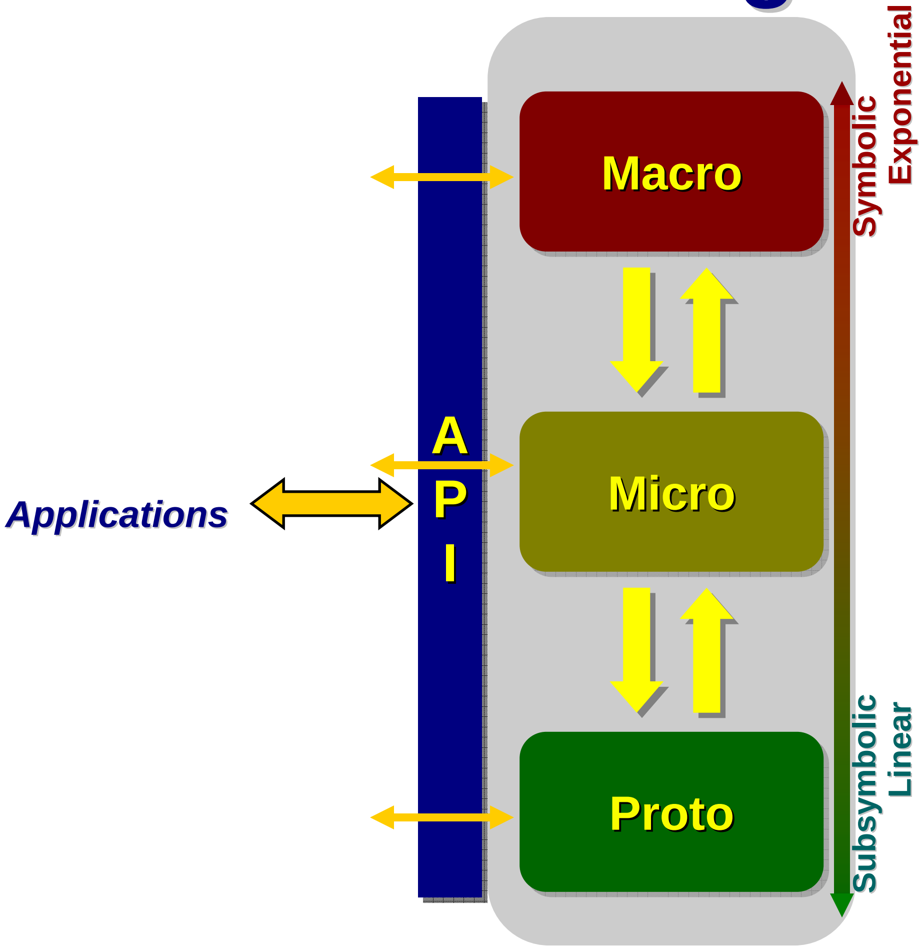
- **The only known general-purpose architecture for solving AI-hard problems**
- **Humans don't solve AI-hard problems through exhaustive search or algorithmic computations**
- **They use (slow but) massively parallel hardware to:**
    - perceive the problem, represent it in structured form and generate perceptual solution constraints
    - apply accumulated expertise to quickly generate local piecewise solutions and combine them into complete solutions
    - apply general knowledge and reasoning ability to refine solutions and overcome holes in expertise
- **They improve their solutions iteratively and dynamically rather than strive for perfect, optimal first-time solutions**



These 3 stages roughly correspond to proto-, micro- and macro-cognition in our architecture

PCAA Cognitive Architecture Design

- **PCAA Cognitive Architecture**
  - Inspired by human cognitive architecture
  - Solves problems in layers:
    - **Proto: Self-organization (perception) layer; Subsymbolic; parallel; O(n)**
    - **Micro: Expertise-based, reactive-reasoning layer; Hybrid; mixed flow; O(n_1)**
    - **Macro: Knowledge-based reasoning layer; Symbolic; serial; O(2^n_2)**
- **Designed for efficient satisficing of hard problems through controlled mix of**
  - symbolic/subsymbolic reps
  - serial/parallel flows

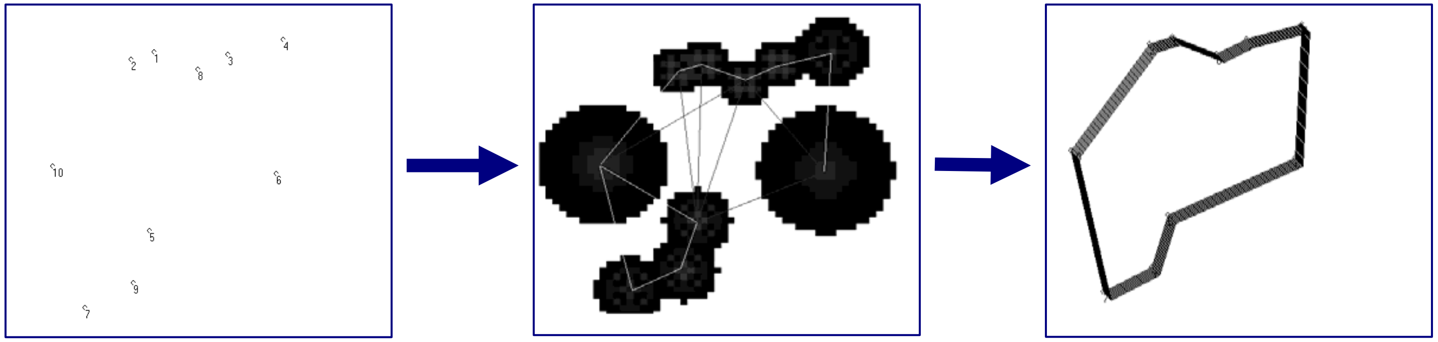


Intelligent collaboration of heterogeneous components to efficiently and robustly solve cognitive problems

Human Solutions to Tour Problem

- **Linearly scalable solution times**
  - Initial moves are more expensive than (constant) subsequent moves
  - Typical human solution within 5% of the optimal solution (good enough?)
  - Profitably ignores non-local high-frequency problem features
- **Human approach: a combination of**
  - global parallel perceptual processes for grouping by proximity (problem decomposition)
  - perception of contours and goodness of path
  - local serial search process

- **People solve a much simpler problem than computer scientists**
- **If the perceptual representation is powerful enough, cognition can be relatively simple and provide good results**
- **Serial search but controlled, adaptive, knowledge-based**



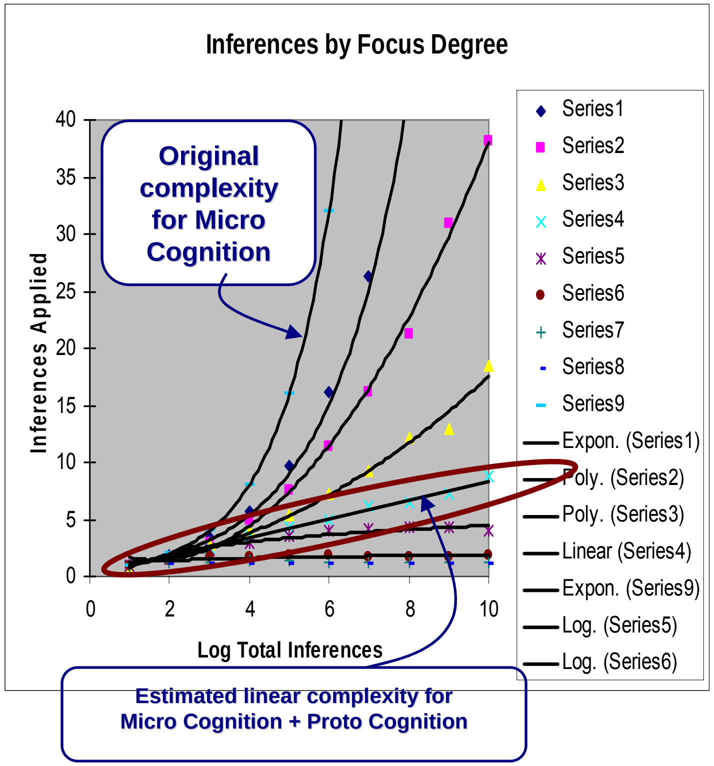
Given the raw problem (left), an intermediate representation based on clustering provides a reduced planning space which the planner rapidly and easily solves

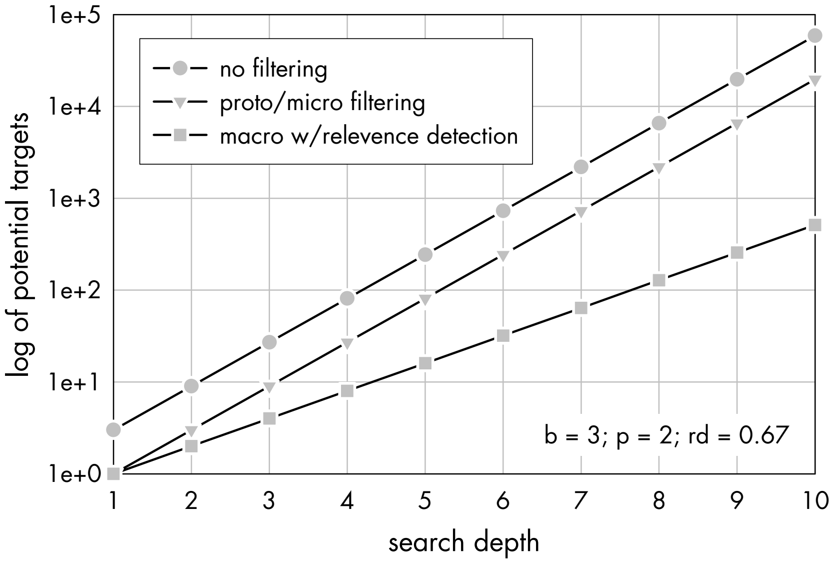
Note: Data from Brad Best's thesis

Some Tentative Results

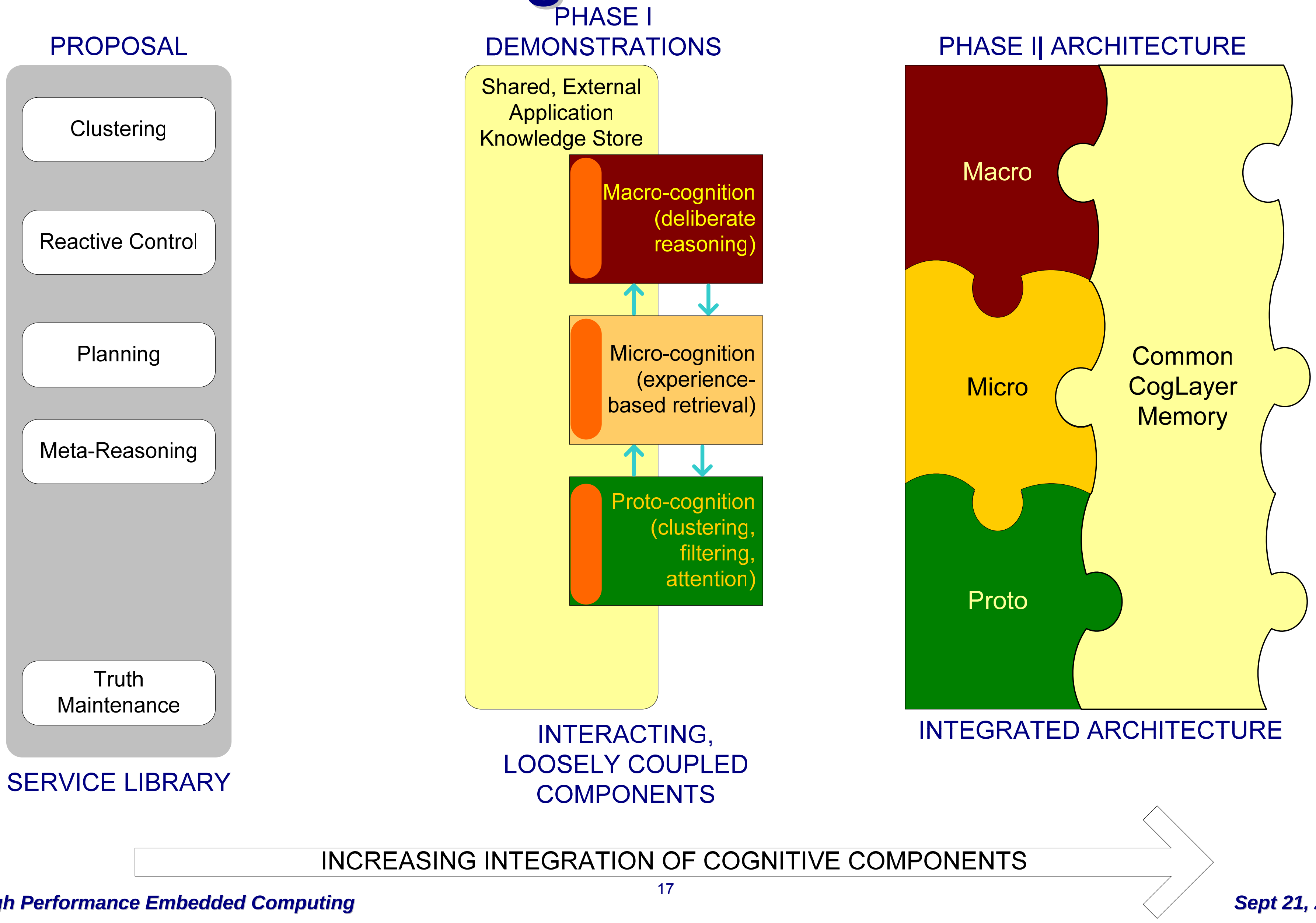
- **Controlled coverage across**
  - symbolic/subsymbolic reps
  - serial/parallel flows

- **Hints of synergistic interactions among layers**
  - Collaboration to reduce individual complexities of cognitions
  - Hints of reduced effective problem-solving complexity





Plan: Maturing Architectural Vision


PROPOSAL
Clustering
Reactive Control
Planning
Meta-Reasoning
Truth Maintenance
SERVICE LIBRARY
PHASE I DEMONSTRATIONS
Shared, External Application Knowledge Store
Macro-cognition (deliberate reasoning)
Micro-cognition (experience-based retrieval)
Proto-cognition (clustering, filtering, attention)
INTERACTING, LOOSELY COUPLED COMPONENTS
PHASE I| ARCHITECTURE
Macro
Micro
Proto
Common CogLayer Memory
INTEGRATED ARCHITECTURE
INCREASING INTEGRATION OF COGNITIVE COMPONENTS

Towards Cognitive Approach

- **Many combinatorial problems can be solved by meta heuristics**
  - Iterated Local Search
  - Tabu
  - Genetic Algorithms
  - Simulated Annealing
  - …
- **Draw on additional heuristics coming from cognitive systems**
  - SOAR
  - ACT-R
  - Swarming
- **Complexity Reduction**
  - Use cognitive techniques for assignment problems as well as approximations for the solvers
  - Intelligently forego exact solutions for timely satisfactory solutions

Mapping the Architecture to Hardware

- **2-pronged strategy**
  - Build kernels for special-purpose functions
    - **Minimum-cost path solver, constraint satisfaction, assignment problems, TSP Solver, etc.**
    - **Payoff in performance of specialized function may justify cost of constructing a kernel**
  - Build kernels for cognitive layer functions
    - **Proto:**
      - Focus: parallelization of front-ends for Micro and Macro
      - Cognitive pyramid (reduce lots of data to small sub-problems)
    - **Macro/Micro**
      - Focus: Speed up “inner loops”
      - Parallelization, customized hardware, etc.
- **Any real-world application will likely draw from both pools of kernels**
  - Long-term goal: Fuse algorithmic and cognitive approaches

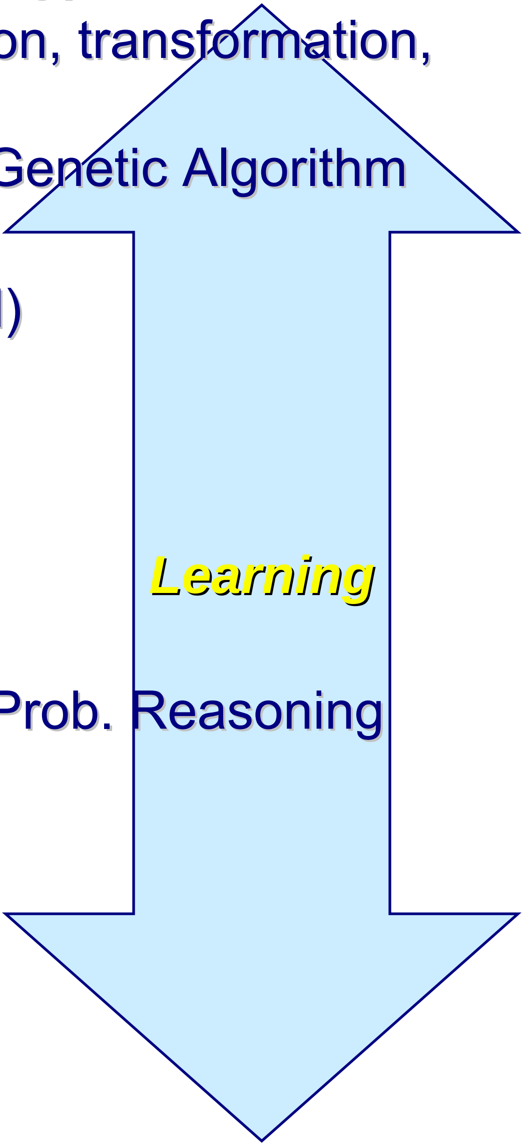
Enabling Hardware Architectures

Some Cognitive Kernels

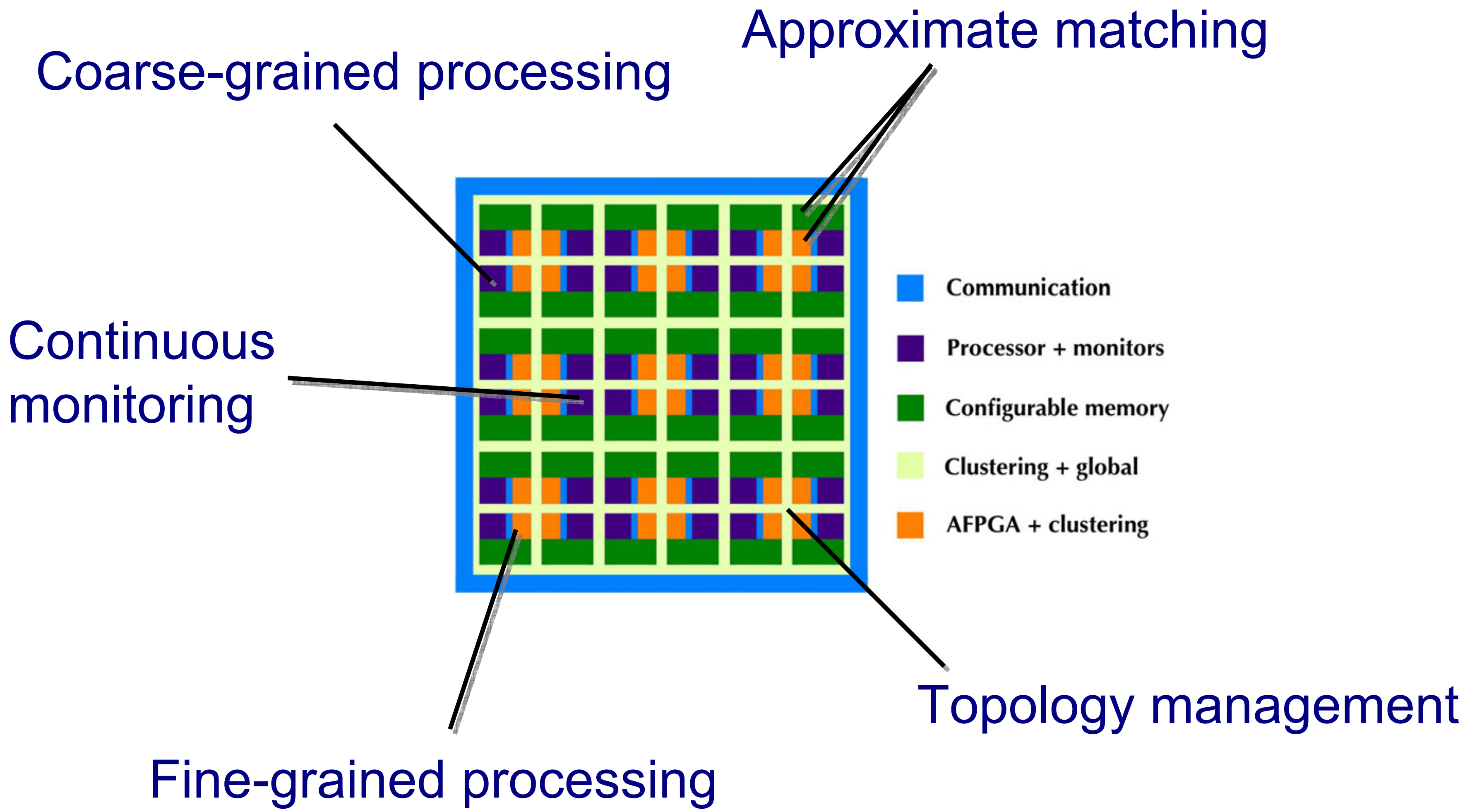
- Minimum-cost path solver
- Constraint satisfaction
- Assignment solver
- Clustering
- TSP solver
- Knowledge-based reasoner
- Probabilistic reasoning
- …

Deriving Hardware Requirements

- **Proto (Swarming, Fine Grained Parallel Sensory)**
  - Acts as perceptual processing (Order reduction, transformation, focus,…)
  - Example Kernels: Clustering, Shortest path, Genetic Algorithm
- **Micro (e.g., ACT-R)**
  - Reactive problem solving (Expertise, retrieval)
  - Serves as interface
    - **Learns and recalls patterns from proto**
  - Develop sub-problem solutions
    - **Transform**
    - **Recall**
  - Example Kernels: Fuzzy Associative Recall, Prob. Reasoning
- **Macro (e.g., SOAR)**
  - Reasoning Engine
    - **High level problem simplification**
  - Explanation based learning
  - Search coordination, Path Search
  - Example Kernels: Automated Reasoning

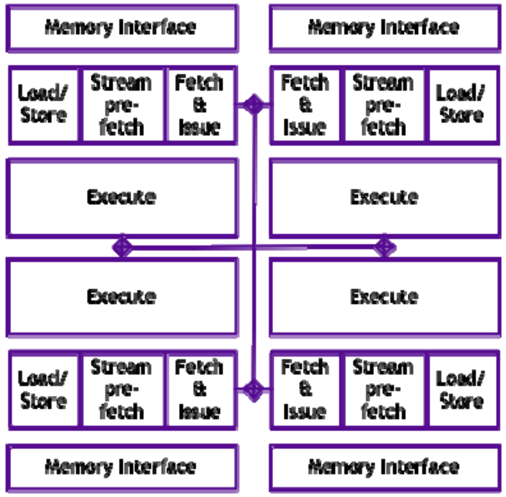


CHASM: Putting it Together



Building Blocks

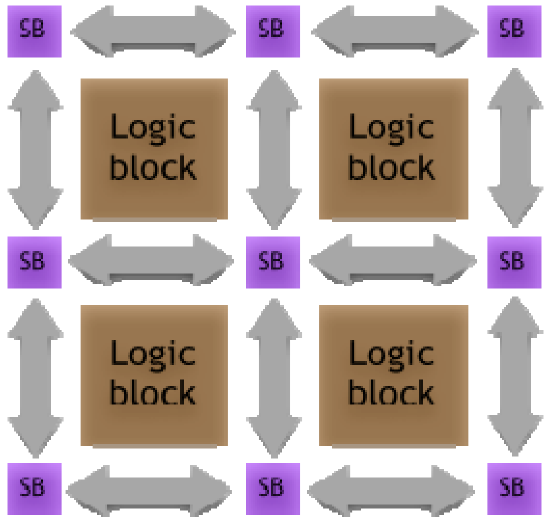
- **Microprocessor**
  - AVM runtime and monitoring
  - Flexibility and generality
  - Coarse-grained processing
    - **SOAR search tasks**
    - **Complex proto-cognition engines**
    - **Rete action rules**



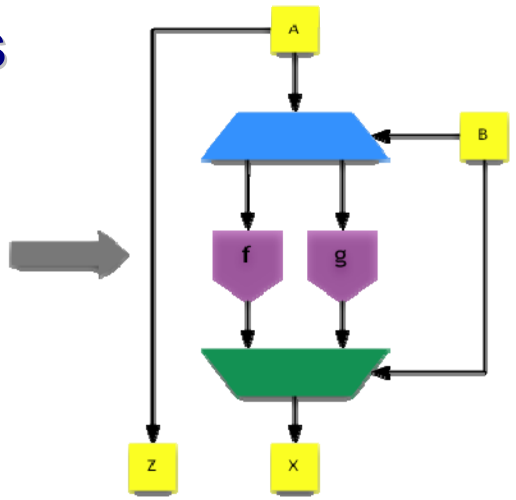
- **Simple, high-frequency**
  - Statically scheduled
  - Multiple execution units
  - Execution bus, memory interface ports to the FPGA
    - **Complex data flows can be statically configured**
    - **E.g. results from memory selected by secondary criteria configured in the FPGA**

Asynchronous Logic

- **Dataflow asynchronous logic**
  - Pipelined lookup tables
  - Pipelined static cross-points
  - Dataflow model of computation
  - 670MHz in .18μm
  - 1.9 GHz in 90nm
  - Support via standard synthesis tools
- **Clustering for reconfiguration**
  - Reduce synchronizations
  - Arbitrary grouping of AFPGA elements possible



```
while (1) {
  rcv(A,a);
  rcv(B,b);
  if (b) {
    x:=f(a);
  } else {
    x:=g(a);
  }
  send(X,x);
  send(Z,a);
}
```

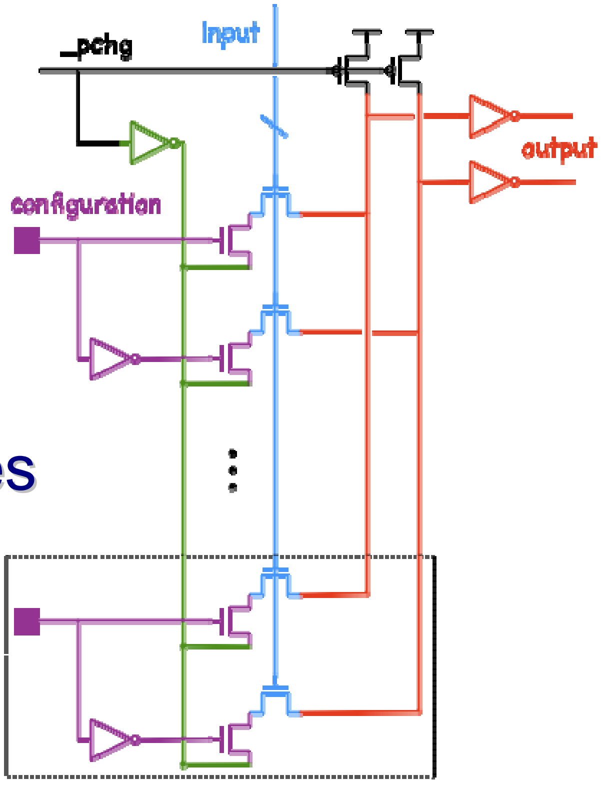


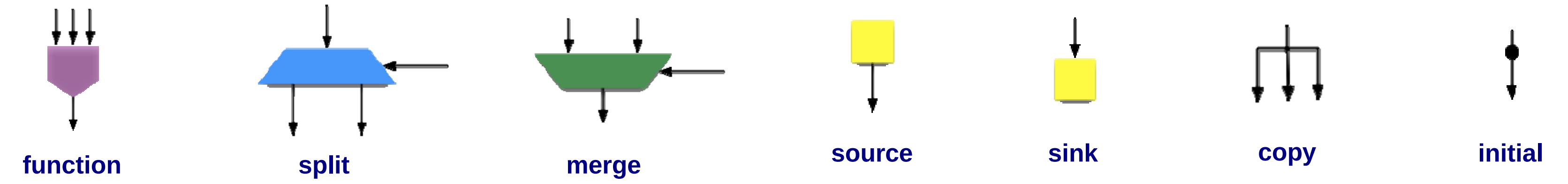
Fast Programmable Logic

- **Conventional: Primary source of performance loss**
  - Poor interconnect performance
  - Configurable clock frequency
  - General logic mapped to mux-based lookup tables
- **Conventional: State-of-the-art**
  - 65nm Virtex-5, max frequency 550 MHz
  - 65nm microprocessors > 3 GHz
  - Standard cell ASICs in 90nm ~ 500-800 MHz
- **Our approach**
  - Use an event-driven model for the FPGA
  - Pipeline interconnect
  - Remove clock network from the core fabric

Fast Programmable Logic

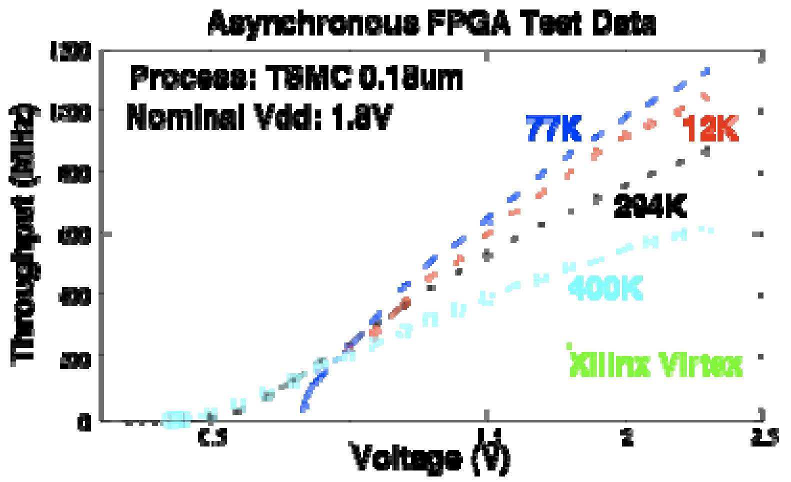
- **Event-driven model**
  - Permits *transparent* pipelining
  - Existing designs map without modification
- **Pipelined Architecture**
  - Enables aggressive circuit families
  - High frequency operation
  - Short circuit paths
  - Our design: 14-18 FO4 cycles



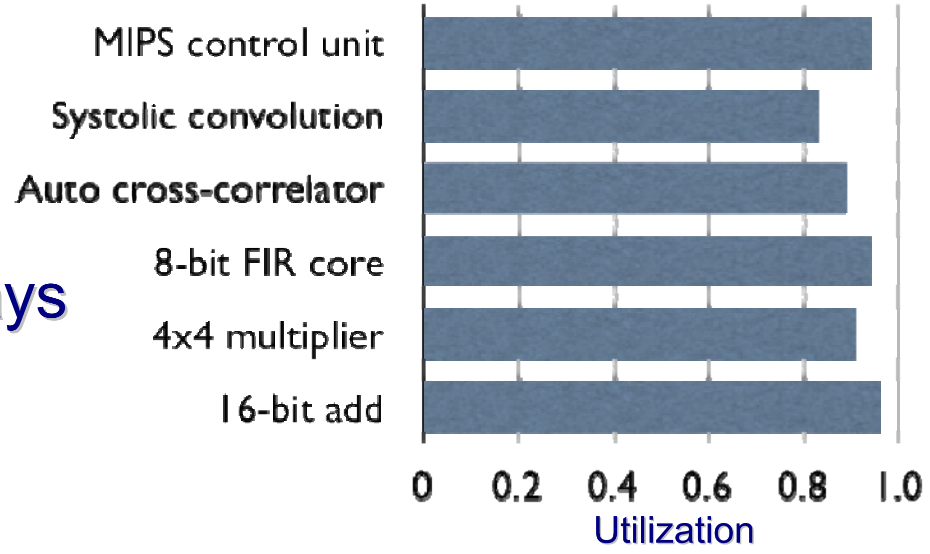


Fast Programmable Logic

- **Robust operation over voltage and temperature**
  - Operates sub-threshold
  - Event-driven has lower power consumption than the synchronous counterpart

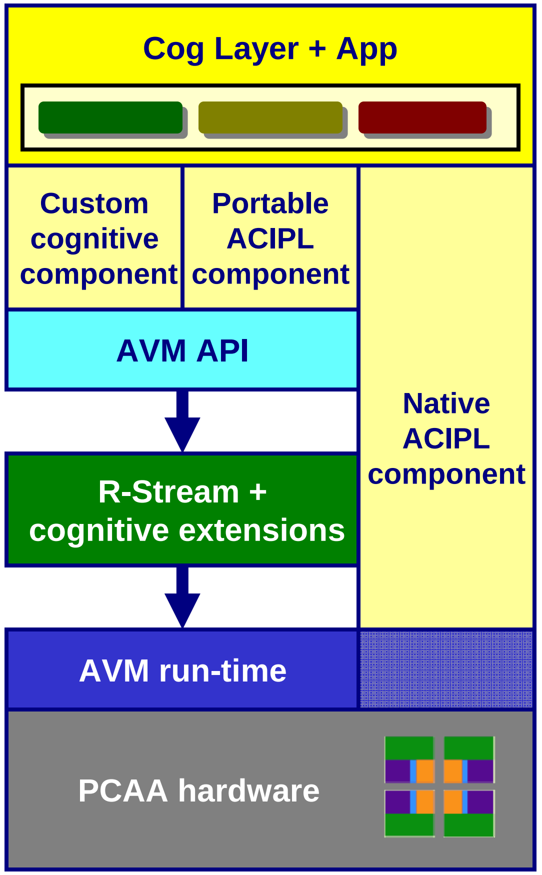


- **2.5x raw performance**
  - High utilization
  - Resilient to wire delays



Cog-to-hardware interface

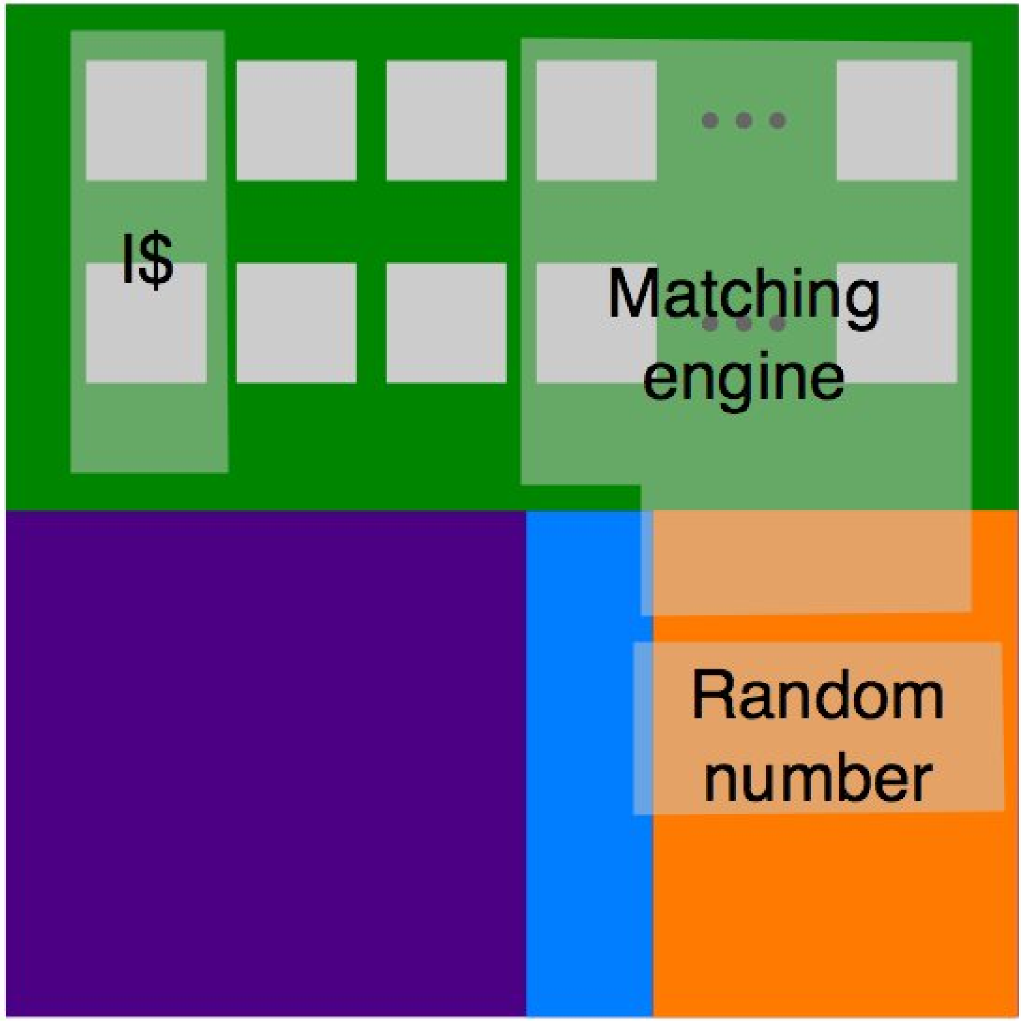
- **Multiple abstraction paths**
  - Native ACIPL: maximum performance, hardcoded into PCAA
  - Portable ACIPL: intermediate performance, predefined but portable
  - Custom to AVM API: user-controlled performance, extensible and portable
- **Runtime system**
  - Load-balancing, grain-size adjustment



Use Library Model to Speed Kernels

- **“Standard” usage patterns**
  - Cognitive engine templates
    - **Proto-cognition: “graph + local update function”**
    - **Micro-cognition: “match operation”**
    - **Macro-cognition: “Rete structure”**
  - Mostly pre-defined configurations plus small flexible part
- **Patterns based on ACIPL library**

Fast Programmable Logic Usage



Memory Considerations

- **Each processing center has parallel banks**
  - Programmable logic in read, write, address paths
  - Inline address transforms
    - **Hash maps**
    - **Sizeable addressability**
    - **Strides, translations**
  - Inline read transformations
    - **Scalar-vector operations**
    - **Filter**
    - **Programmable Metric**
      - Operate with scalar operand
      - Programmable variable (inc,dec,rand)
  - Inline write transformations
    - **Scale, offset, filter**

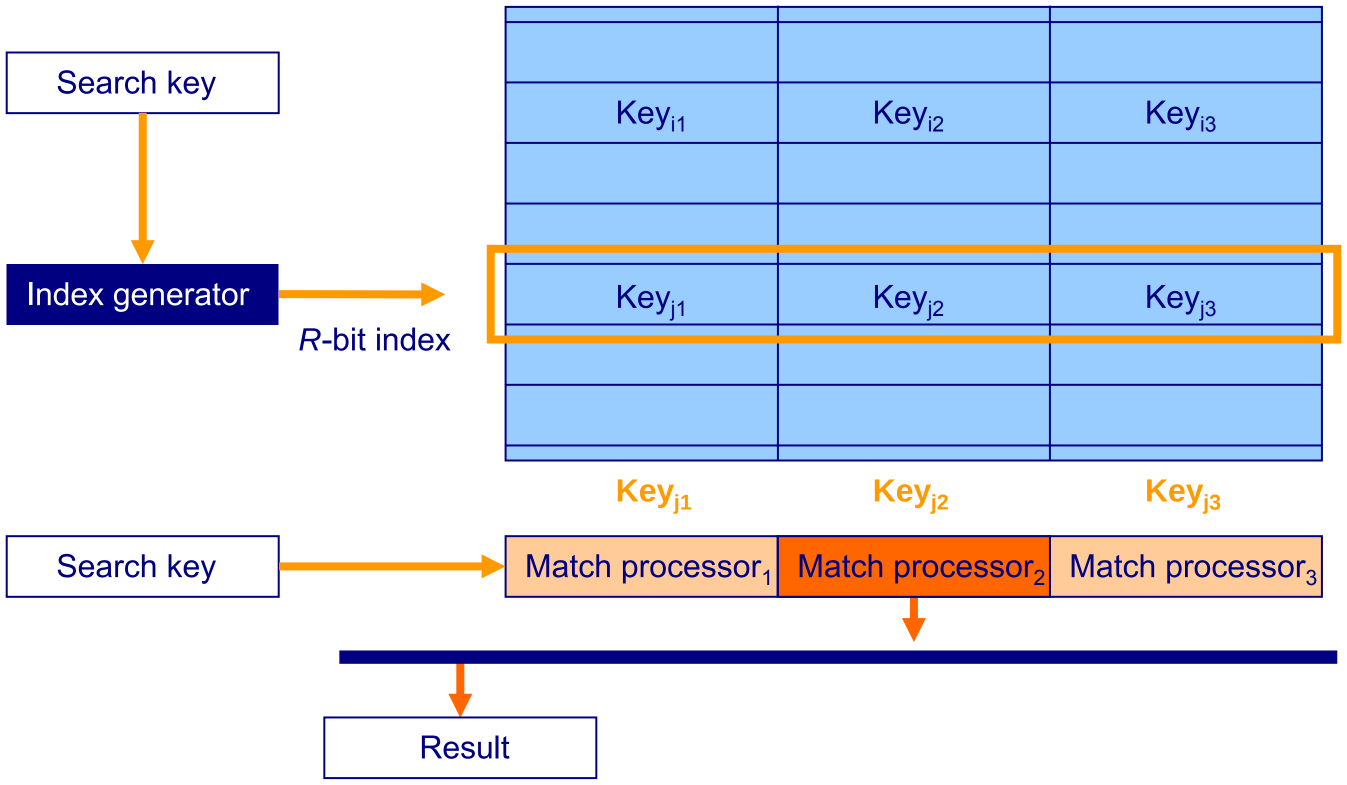
Memory Considerations

- **Hashing – a software solution**
  - Fastest software technique for large databases
  - Multiple memory accesses (slow)
  - Cache pollution
- **CAM (Content Addressable Memory) – a hardware solution**
  - High bandwidth
  - Large area
  - Power-inefficient

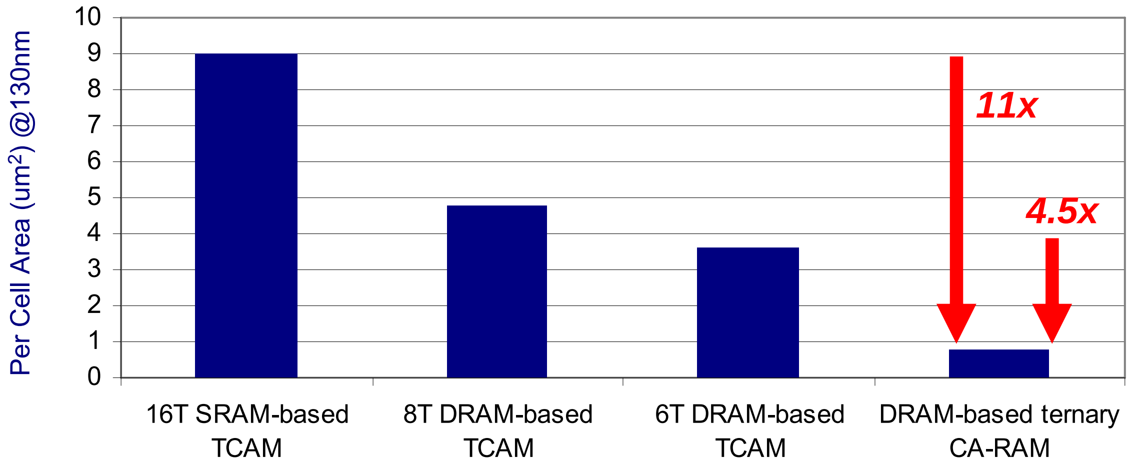
- **CA-RAM (Content-Addressable RAM)**
  - High bandwidth
  - Area-efficient
  - Power-efficient
  - Hash mapped to APL in address path
- **POEM (Programmable Objective Evaluation Memory)**
  - Parallel closest match
  - Activation Calculus boost implemented APL
  - Resolution implemented in APL

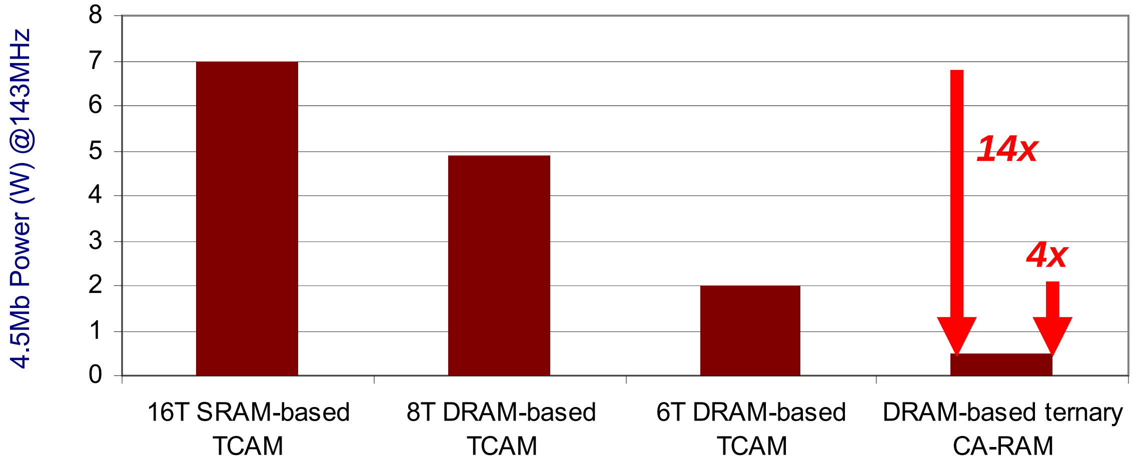
Our experiments

Memory: CA-RAM Operation



Memory: CA-RAM Critical Benefits





Metrics

What are success metrics?

- **Basic**
  - SWEPT to formulate plan
- **Mission Performance**
  - Initial and dynamic replan time
  - Mission flexibility (# automatically planned / # total types)
  - Planning detail (# of parameters determined automatically / # desired)
  - Planning simultaneity
  - Contingency management (# handled/#desired)
  - Plan quality (#constraints violated/#total)
  - Team collaboration (#automated assets/#total)

Metric Considerations (cont.)

- **System performance**
  - Throughput / Latency
  - Plan Fidelity (Parameters available/Parameters Used)
  - Plan Latency / Plan Fidelity Quotient
  - Available task allocation
  - Signature management utilization
  - Exposure time
  - Asset utilization rate

Metric Considerations (cont.)

- **System stability**
  - Plan generation exception rate
  - Information loss / corruption rate

- **Software Usability**
  - Operator/Pilot Workload
  - Software ease of use
  - Situational awareness

Conclusion

Recap

- Autonomous Vehicles demand new approaches for algorithm design and computation
- We are experimenting with cognitive techniques to address limitations of “algorithmic techniques”
- We are developing new system and hardware techniques to improve the performance and applicability of the “cognitive techniques”