An Ecological Forecasting Agent


3/28/2005
Project goals

• Make data analysis faster and cheaper
• Increase use of NASA data by removing barriers to data access
  – Cope with data heterogeneity
• Support code reuse and rapid application development
• Support multiple applications, users
  – Including fire and health domains
• Improve QOS
  – Always provide an answer
  – Tell user how good it is, where it came from
Team members

- Keith Golden -- PI, Planner
- Wanlin Pang -- Constraint Reasoning
- Rama Nemani -- Science
- Petr Votava -- TOPS implementation
- Oren Etzioni -- Natural Language Interface
- David Danks -- Automated discovery of models
- Forrest Melton -- Project management
- Andy Michaelis -- TOPS
Agents

• An agent is an intelligent assistant
  – E.g., travel agent

• Provides *goal-oriented* interface
  – You say *what* to do, not *how* to do it.
  – Agent has the knowledge to figure out *how*.

• Copes with uncertainty and error robustly.
  – Obtains information needed to complete task
  – Tries something else when encountering failure
Terrestrial Observation and Prediction System

Weather Networks
- Temperature/rainfall/radiation/humidity/wind

Orbiting Satellites
- Terra/Aqua/Landsat/Ikonos
  - Landcover/change, Leaf area index, surface temperature, snow cover and cloud cover

Ancillary Data
- Topography, River networks, Soils

Ecosystem simulation models

Weather & Climate Forecasts

Monitoring & Forecasting
- Stream flow, soil moisture, phenology, fire risk, forest/range/crop production
  • 3 research assistants for 12 months
  • processed <15 GB of data
  • data preparation >80%

• EOSDIS:
  • Generates ~3 tera-bytes of data a day.
  • Currently holds 2 peta-bytes
  • 1 day = 2 years of HUBBLE Space Telescope
Dimensions of Autonomy

• Range of controllable behavior
  – Automate the generation of the forecasts, analysis of the results, and model adjustments
  – Generate and record “meta-data” to facilitate later searches
  – Integrate of new models and data sources in “plug & play” fashion

• Command specificity
  – High-level goals (descriptions of desired data products)
  – No premature commitment to specific format, resolution, projection, etc.

• Execution Robustness
  – Adapt to changes and recover from failures
  – Out of several sources of the same data use the “best” available one
Distributed Agent Architecture

User Interface
- Expert UI
- NLI
- Web Interface

JDAF Core / TOPS
- RMI Interface
- Web Services
- TOPS Models & Filters
- TOPS Database

IMAGEbot
- Database
- Planner (Doppler)
- Executive
- Task Manager
- JNET

Nat'l. Data Centers
- UW PRECISE
- UMT TOPS
- NASA Ames IMAGEbot
- Fetch, Theseus (Marina Del Rey, CA)

Ames Research Center
3/28/2005
Planner capabilities

• Automate data collection and processing
• Integrate multiple data sources
  – Now: satellite, weather stations
  – Eventually: UAV, ground sensors
• Support flexibility and robustness
  – Constraints allow consistency without premature commitment to data source, file format, resolution, projection.
  – Preferences allow planner to produce best data possible
    • According to user’s priorities
    • Subject to available data sources
  – Multiple options + backtracking means errors or dropouts won’t result in failure.
• Ease application development through abstraction
Data Processing Action Description Language

• Declarative representation of data sources, models and data product specifications
  – No need to specify scripts to process data
  – Modular, plug & play design

• Actions describe data-processing operations
  – Earth system models, data filters, composition, etc.
  – Any number of inputs and outputs

• Arbitrary constraints over any static type

• Integration with Java
  – Embedded Java code
    • Integration with JDAF or other systems
    • Action execution
    • “Procedural” constraints
  – Parameters include Java objects
Desired “planning language” properties

• Easily useable by planners
  – Declarative semantics
  – States, actions
  – As feature-poor as possible

• Compatible objectives?
  – Commonality: state, state change
    • Variables
    • Actions ≈ Procedures/methods
    • Preconditions ≈ tests
    • Conditional effects ≈ conditional instructions
  – “Compile” to simpler language
DPADL: Data Processing Action Description Language

- Object oriented, C++/Java-like syntax
  - Inheritance
  - Primitive types and objects
  - Object creation, copying, modification

- Integration with Java
  - Embedded Java code
    - Action execution
    - “Procedural” constraints
      - Parameters include Java objects

- Actions describe data-processing operations
  - Any number of inputs and outputs
  - Causal, declarative representation of data filters

- Constraints over any static type
Types

- Can inherit from objects or primitive types
  ```java
  static type Filename instanceof String;
  ```
- Can be defined by list of members
  ```java
  static type ImageFormat =
  {"JPG", "GIF", "TIFF", "PNG", "XCF", ... };

  static type ProjectionType =
  {LAZEA=11, GOODE_HOMOL=24, ROBINSON=21, ... };
  ```
- Can represent complex data structures
  ```java
  type Image instanceof Object {
      static int xSize;
      static int ySize;
      PixelValue pixelValue(int x, int y);

      ...
  }
  ```
Functions

• Can be static or fluent
  ```
  fluent float temperature(float lon, float lat);
  static float sin(float x);
  ```

• Are the atoms of the language
  – Attributes are functions of their objects
  – Infix operators are functions (C++ style overloading)
    ```
    static String operator+(string s1, string s2);
    ```
  – Relations are boolean functions
    ```
    static boolean operator<(float r1, float r2);
    ```
  – Global variables are fluent functions with no arguments
    ```
    fluent Date currentDate;
    ```

• Can be targets of assignment
  ```
  image1.pixelsValue(x, y) := image2.pixelsValue(y, x);
  ```
Constraints

Can be attached to types or functions

```java
static type Filename isa string {
    constraint Matches(true, this, "~[/]+")
}
static string operator+ (string s1, string s2) {
    constraint Concat(value, s1, s2);
}
```

Can be specified using embedded Java code.

```java
static type Tile isa object mapsto tops.modis.Tile {
    Instrument product {
        constraint {
            result(this) := $ this.getProduct() $
        }
    }
}
```

3/28/2005

NASA Ames Research Center
Constraints

static type Tile isa object mapsto tops.modis.Tile { ...
  //true if this tile covers the specified location
  boolean covers(float lon, float lat) {
    constraint {
      {this}([lon], [lat], d=day, y=year,
        p=product, value) := {
        if(value)
          return tm.getTiles(lon.max, lat.min,
            lon.min, lat.max,
            d, y, p);
        else return null; }
    }
    ...
  }
Actions

action threshold (unsigned thresh) {
    input BwImage in;
    output BwImage out copyof in;
    forall unsigned x, unsigned y;
    effect when (x < in.xSize && y < in.ySize) {
        when (in.valueAt(x, y) <= thresh) {
            out.valueAt(x, y) = BLACK;
        } else {
            out.valueAt(x, y) := WHITE;
        }
    }
    exec $ out = gfx.threshold(in, thresh); $;
}
Desired “programming language” properties

• Naturally describe domain concepts
  – Data semantics and syntax
  – Structured data files, complex data types
  – Object creation/copying

• Specify interfaces external environment
  – Execute plans
  – Obtain information (sensing)

• Easily usable by programmers
  – Similarity to known programming languages
  – As feature-rich as necessary
Data Goals and Metadata

• Data product specification
  – **What** information is contained
  – **How** information is encoded in data
  – **Where** the data files are stored/delivered
  – **When** the information pertains to

• Examples
  – I want an MPEG movie of yesterday’s weather over the SF bay placed on our website
  – File dd010101.tar.gz is a compressed archive of the downlink directory as of Jan 1, 2001
Planning for data processing

- Initial state = data available
- Goal = data to produce
- Plan = dataflow program
- Actions =
  - Earth system models
  - Data transformations
- Domain characteristics
  - Very large universes
  - Complex data structures
  - Lots of constraints
+ Highly parallel

3/28/2005
Dataflow plans

Inputs

MODIS
FPAR
Daily

MODIS
LAI
Daily

GOES

RUC2
GRIB
WGRIB
bin

Filters

FPAR
LAI

Land
Surface
Models

Models

Drill-
down

Min, Max Temp
Mean Temp
Mean wind

Topography

 Soil Moisture
Snow cover
Stream flow
NPP
Phenology

Visualization

Statistics

False
Color

3/28/2005

NASA Ames Research Center
## Input data choices

<table>
<thead>
<tr>
<th>Data Source</th>
<th>Category</th>
<th>Resolution</th>
<th>Time Frame</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Terra-MODIS</td>
<td>FPAR/LAI</td>
<td>1 day</td>
<td>1km, 500m, 250m</td>
<td>global, since 2000</td>
</tr>
<tr>
<td>Aqua-MODIS</td>
<td>FPAR/LAI</td>
<td>1 day</td>
<td>1km, 500m, 250m</td>
<td>global, since 2002</td>
</tr>
<tr>
<td>AVHRR</td>
<td>FPAR/LAI</td>
<td>10 day</td>
<td>1 km</td>
<td>global, since 1981</td>
</tr>
<tr>
<td>SeaWIFS</td>
<td>FPAR/LAI</td>
<td>1 day</td>
<td>1m x 4 km</td>
<td>global</td>
</tr>
<tr>
<td>DAO</td>
<td>temp, precip, rad, humid</td>
<td>1 day</td>
<td>1.25 deg x 1.0 deg</td>
<td>global, since 1980</td>
</tr>
<tr>
<td>RUC2</td>
<td>temp, precip, rad, humid</td>
<td>1 hour</td>
<td>40 km</td>
<td>USA</td>
</tr>
<tr>
<td>CPC</td>
<td>temp, precipitation</td>
<td>1 day</td>
<td>point data</td>
<td>USA</td>
</tr>
<tr>
<td>Snotel</td>
<td>temp, precipitation</td>
<td>1 day</td>
<td>point data</td>
<td>USA</td>
</tr>
<tr>
<td>GCIP</td>
<td>radiation</td>
<td>1 day</td>
<td>0.5 deg</td>
<td>Continental</td>
</tr>
<tr>
<td>NEXRAD</td>
<td>precipitation</td>
<td>1 day</td>
<td>4 km</td>
<td>USA</td>
</tr>
</tbody>
</table>

3/28/2005

NASA Ames Research Center
Complex Data Structures

![Diagram of complex data structures with Tables and Relationships]
Planning as state-based search

- Planning problem: \(< s_0, \mathcal{A}, \mathcal{G}>\)
  - \(s_0\) = Initial state
  - \(\mathcal{A}\) = Set of actions (state transforms)
  - \(\mathcal{G}\) = set of goal states
- Plan: sequence of actions from \(s_0\) to state \(s_n \in \mathcal{G}\)
Planning as Data Production
(First Draft)

- Planning problem: $< D_0, A_D, G_D >$
  - $D_0$ = Initial data
  - $A_D$ = Set of actions (data transforms)
  - $G_D$ = set of goal data
- Plan: sequence of actions from $D_0$ to state $D_n \in G_D$

3/28/2005

NASA Ames Research Center
Planning as Data Production

• Planning problem: \(<I_D, A_D, G_D>\)
  – \(I_D\) = Set of Initial data
  – \(A_D\) = Set of actions (data transforms)
  – \(G_D\) = set of goal data sets
• Plan: DAG of actions from \(I_D\) to \(D_n \in G_D\)

3/28/2005
NASA Ames Research Center
Planning approach

1. Perform graph analysis to derive
   - *distance heuristics*
   - *initial variable domains*
2. Convert planning problem into CSP
3. Search for solution using heuristics
   - Sensors represented as constraints
4. Execute plan, update database and replan if needed
Preferences

- Not all “satisfactory” data products are equally good
- Constraints only allow us to specify what is (un)acceptable, not what is preferred
- Preferences allow users to specify what matters to them (resolution, quality, speed)
Optimality

• IMAGEbot can produce the best possible data product given user preferences
  – Optimization handled as part of constraint search
  – Computationally hard, could take a while for some problems

• Pareto Optimality
  – When multiple unordered preferences are given
    • I want the best quality and I want it now
  – No preference could be satisfied more without satisfying another less
    • I can’t give you better quality without spending more time
    • I can’t give you a faster answer without sacrificing quality
Problem graph analysis

- Obtain early information on
  - Possible parameter values
  - Appropriate data sources
  - Tradeoffs among optimization criteria
Constraints as glue

Dataflow Plan

Constraint Network

Runtime Environment (JVM)
Natural Language Interface

Build on Precise NLIDB

1. English → SQL
2. Resolve ambiguity
3. SQL → DPADL
4. Resolve conflicts
5. Planner solves goal
6. User updated with result
PRECISE queries

- What is the area of Alaska?
- What is the population of New York?
- Which are all the states that border Oregon?
- How many major cities are in Florida?
- How long is the Colorado River?
- What rivers traverse Indiana and Illinois?
- What rivers traverse Indiana or Illinois?
- What cities are in Texas and have a population of less than 100000 people?
- What is the largest city in the smallest state in the US?
- What states border the state with the largest population?

http://www.cs.washington.edu/research/projects/WebWare1/www/precise/precise.html
Comparison to state of art

Fraction Answered

Recall = \( \frac{Q_{\text{answered}}}{Q} \)

Recall > 75%

Error Rate

Precision = \( \frac{Q_{\text{correct}}}{Q_{\text{answered}}} \)

PRECISE made no mistakes on semantically tractable questions
Publications


Related Work

• GENESIS
  – SciFlo similar to JDAF
  – Doppler planner could be used to generate SciFlow “plans”

• IDACT
  – Automatic data type transformations
  – Restricted form of planning for dataflow generation

• DISCOVER
  – Transparent data access
  – Use of ESML

• GeoBrain
  – A geo-tree is a dataflow plan
  – No support (yet) for automatic generation
Related Work (cont’d)

• MVP, COLLAGE
  – Scientific image processing, human in the loop
  – HTN representation, less need for precise causal representation
  – No causal reasoning about data, metadata generation

• Chimera
  – Data tracking, but no support for causal reasoning

• Amphion, AutoBayes
  – Program synthesis using theorem proving
  – More expressiveness than needed for many DP problems

• Internet Softbot
  – Information gathering and changes to world.
  – Could never handle Unix pipes
Daily Ecocast

Daily GPP 6/20/2004

What is Ecocasting?

Ecological forecasting (or ‘ecocasting’) is the prediction of ecosystem parameters. NASA Ames is developing advanced computing technologies for converting massive streams of satellite remote sensing data into ecocasts that are easy to read and use.

NASA Ames, UWF IHMC, CMU, CSUMB, UMT, UW, and Fetch Technologies are collaborating to develop a distributed computing architecture for the production of ecocasts from satellite remote sensing data and other ancillary data sources. Applications of the Ecocast technology include fire forecasting, crop quality forecasting, snowpack and flood monitoring, and identification of anomalies in the carbon cycle and other biospheric processes.

News

Daily updates of biospheric parameters are now available. See below for a selection of available parameters. Or download data and images here.

Nowcasts & Forecasts

- Meteorology
- Hydrology
- Carbon Cycle