Diagnosing Automatic Whitelisting for Dynamic Remarketing Ads Using Hybrid ASP

Alex Brik[1] and Jeffrey B. Remmel[2]

LPNMR 2015

September 2015

[1]Google Inc
[2]UC San Diego

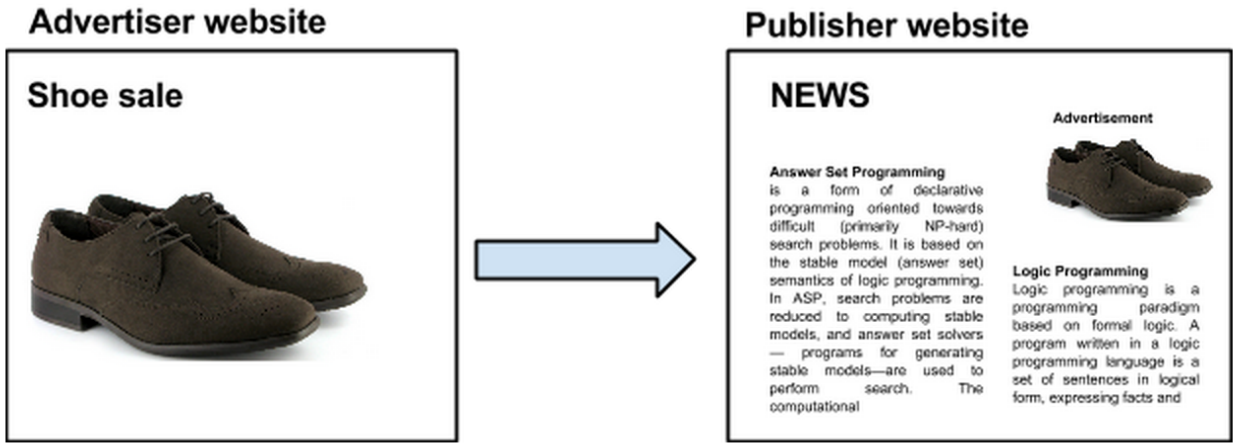
Hybrid ASP (H-ASP) is an extension of ASP for combining ASP type rules and numerical algorithms. Dynamic Remarketing Ads is Google's platform for showing customized ads based on past interactions with a user. The presentation will descibe the use of H-ASP in creating software that diagnoses possible reasons why an advertiser is not ready to show dynamic remarketing ads.

Outline:

Automatic Whitelisting for Dynamic Remarketing Ads

- In order for Google to start showing dynamic remarketing ads for a particular advertiser the readiness of that advertiser needs to be verified (advertiser needs to be whitelisted)
- Readiness is verified automatically
- There are multiple ways for an advertiser to prepare for showing dynamic remarketing ads
- In every case a set of constraints needs to be satisfied
- Satisfaction of a constraint may depend on the contents of logs, ads databases and data from the previous constraint evaluation

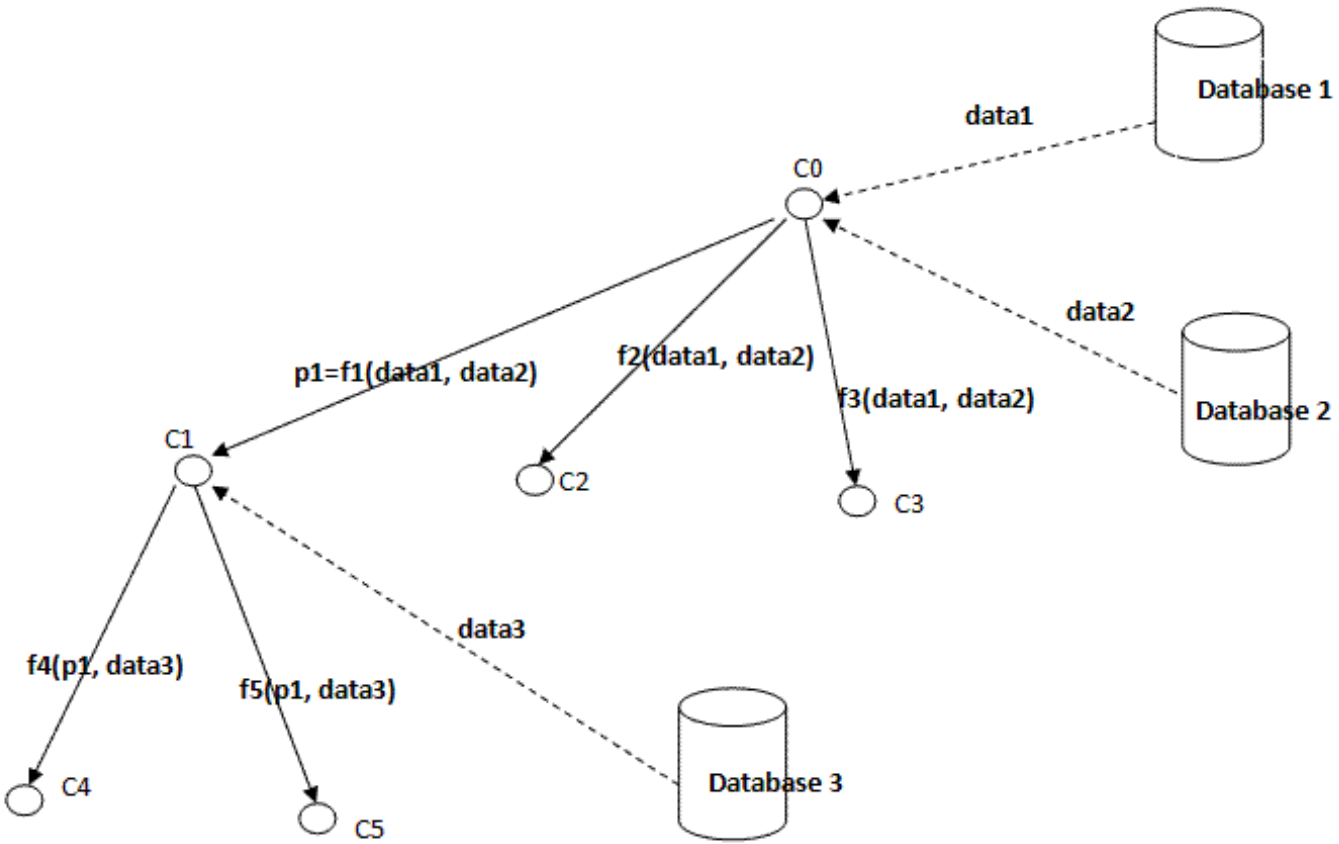
Automatic Whitelisting for Dynamic Remarketing Ads



An example of advertiser becoming ready to show dynamic remarketing ads:

- An advertiser needs to install a JavaScript tag on their website (verified by searching advertiser related database), and
- A user visits advertiser's website (verified by searching a user event log), and
- An advertiser creates a dynamic remarketing ad (verified using a function from a separate software library)

Automatic Whitelisting for Dynamic Remarketing Ads

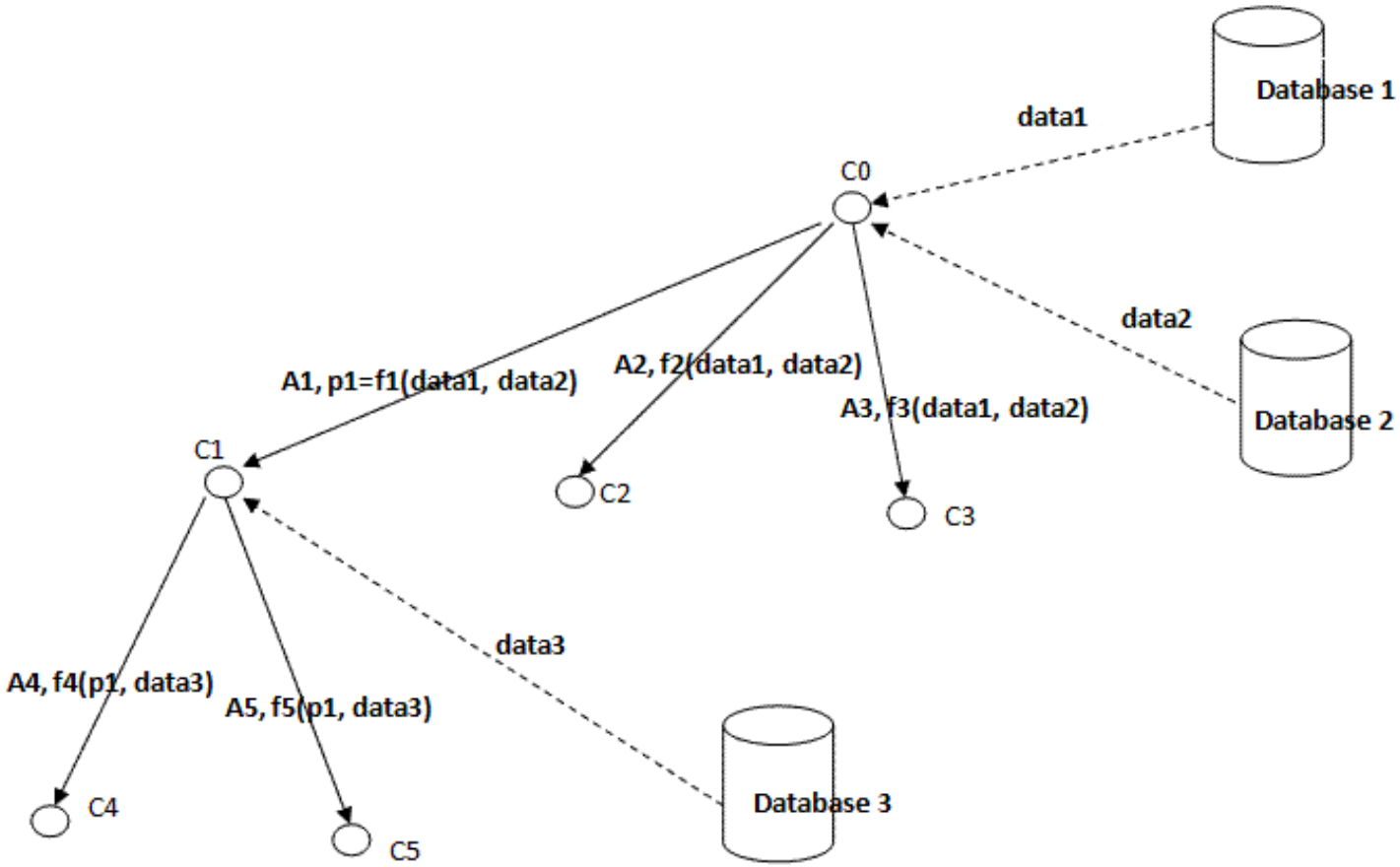


- A path from the root to a leaf with all constraints verified results in an advertiser being whiltelisted.
- A diagnosis contains an unsatisfied constraint for every case

Branching Computational Pattern (BCP)

- Diagnosing a failure of automatic whitelisting can be viewed as an instance of the Branching Computational Pattern (BCP)
- BCP is a pattern of computation for generating a connected directed acyclic graph (cDAG) with the set of vertices consisting of pairs $(A, \mathbf{p})$, where A is a set of atoms and $\mathbf{p}$ is a vector of parameter values representable by a computer.
- At each cDAG vertex $(A, \mathbf{p})$, the computation consists of the two steps
  1. use A and $\mathbf{p}$ to choose algorithms (that will possibly access external data repositories and/or perform computations) to produce the set of next parameter value vectors $\mathbf{q}_1, ..., \mathbf{q}_k$ and
  2. for each $\mathbf{q}_i$ produced in step 1, derive sets of atoms $B_{i,1}$, $B_{i,2}$, ..., B_{i,m_i}.
- The set of child states of $(A, \mathbf{p})$ is $\{(B_{1,1}, \mathbf{q}_1)$, $...,(B_{1,m_1}, \mathbf{q}_1)$, $(B_{2,1}, \mathbf{q}_2)$, $...,(B_{k,m_k}, \mathbf{q}_k)\}$.
- The cDAG generated by a BCP with the sets of atoms restricted to At and the set of vectors of parameters restricted to S is called a **computational cDAG over At and S**.

Branching Computational Pattern (BCP)



Hybrid Answer Set Programming (H-ASP)

H-ASP can be used to compute according to BCP.

- H-ASP is an extension of ASP for combining logical reasoning and numerical processing
- A program P has a parameter space S and a set of atoms At.
- Elements of S (*generalized positions*) are of the form $\mathbf{p} = (t, x_1, ..., x_m)$, t is time and x_i are parameter values, notation: $t(\mathbf{p}) = t$ and $x_i(\mathbf{p}) = x_i$.
- The universe of P is $At \times S$
- For $M \subseteq At \times S$, $\widehat{M} = \{\mathbf{p} \in S : (\exists a \in At)((a, \mathbf{p}) \in M)\}$ - the set of generalized positions used in M
- For $\mathbf{p} \in S$, $W_M(\mathbf{p}) = \{a \in At : (a, \mathbf{p}) \in M\}$ - the set of atoms corresponding to $\mathbf{p}$ in M. $(W_M(\mathbf{p}), \mathbf{p})$ is a **hybrid state**.
- A **block** B is of the form $B = a_1, ..., a_n, not\ b_1, ..., not\ b_m$

$$M \models (B, \mathbf{p})$$

if $\mathbf{p} \in \widehat{M}$ and $(a_i, \mathbf{p}) \in M$ for $i \in \{1, ..., n\}$ and $(b_j, \mathbf{p}) \notin M$ for $j \in \{1, ..., m\}$.

Hybrid Answer Set Programming (H-ASP)

The rule types are restricted to those relevant for computing with BCP.
Two types of rules: **Advancing rules** (used to derive new generalized positions)

$$a \leftarrow B : A, O$$

where B is a *block* (of the form $B = a_1, ..., a_n, not\ b_1, ..., not\ b_m$), $O \subseteq S$, A is a set valued algorithm $O \longrightarrow 2^S$.
The idea: for $\mathbf{p} \in O$ if $M \models (B, \mathbf{p})$ then $(a, \mathbf{q})$ should be in M for all $q \in A(\mathbf{p}) \cap \widehat{M}$.
Stationary rules (used as constraints and to derive new atoms)

$$a \leftarrow B_1; B_2 : H, O \quad (\text{or} \quad a \leftarrow B : H, O)$$

where each B_i is a block, $O \subseteq S^2$ (or $O \subseteq S$), H is a boolean algorithm.
The idea: for $(\mathbf{p}_1, \mathbf{p}_2) \in O$ with $t(\mathbf{p}_1) < t(\mathbf{p}_2)$ (or $\mathbf{p} \in O$) if $M \models (B_i, \mathbf{p}_i)$ $(M \models (B, \mathbf{p}))$ and $H(\mathbf{p}_1, \mathbf{p}_2)$ $(H(\mathbf{p}))$ is true then $(a, \mathbf{p}_2) \in M$ $((a, \mathbf{p}) \in M)$.

Hybrid Answer Set Programming (H-ASP)

Stable Model

- A **Horn program** does not contain any negated atoms in At.
- For a Horn program P, an initial condition $I \in S$, and $M \subseteq At \times S$, **one-step provability operator** $T_{P,I}(M)$ is M union all (a, J) s.t
  1. $\exists$ a stationary rule $C = a \leftarrow B : H, O$ and $\mathbf{p} \in O \cap \left(\widehat{M} \cup \{I\}\right)$ such that $M \models (B, \mathbf{p})$ and $H(\mathbf{p}) = 1$ and $J = \mathbf{p}$, or
  2. $\exists$ a stationary rule $C = a \leftarrow B_1; B_2 : H, O$ and $(\mathbf{p}_1, \mathbf{p}_2) \in O \cap \left(\widehat{M} \cup \{I\}\right)^2$ such that $M_i \models (B_i, \mathbf{p}_i)$ and $H(\mathbf{p}_1, \mathbf{p}_2) = 1$, and $J = \mathbf{p}_2$ or
  3. $\exists$ an advancing rule $C = a \leftarrow B : A, O$ and $\mathbf{p} \in O \cap \left(\widehat{M} \cup \{I\}\right)$ such that $M \models (B, \mathbf{p})$ and $J \in A(\mathbf{p})$

$T_{P,I}(\emptyset) \uparrow \omega = \bigcup_{k=0}^{\infty} T_{P,I}^{k}(\emptyset)$ is the least model of P with the initial condition I.

Hybrid Answer Set Programming (H-ASP)

Stable Model

- An advancing rule $C = a \leftarrow B : A, O$ is **inconsistent with** (M, I) if for all $\mathbf{p} \in O \cap \left(\widehat{M} \cup \{I\}\right)$ either
  1. $M \not\models (B^-, \mathbf{p})$
  2. $A(\mathbf{p}) \cap \widehat{M} = \emptyset$
- A stationary rule $C = a \leftarrow B_1, B_2 : H, O$ is **inconsistent with** (M, I) if for all $(\mathbf{p}_1, \mathbf{p}_2) \in O \cap \left(\widehat{M} \cup \{I\}\right)^2$ either
  1. $M \not\models \left(B_1^-, \mathbf{p}_1\right)$ or $M \not\models \left(B_2^-, \mathbf{p}_2\right)$ or
  2. $H(\mathbf{p}_1, \mathbf{p}_2) = 0$
- A stationary rule $C = a \leftarrow B : H, O$ is **inconsistent with** (M, I) if for all $\mathbf{p} \in O \cap \left(\widehat{M} \cup \{I\}\right)$ either
  1. $M \not\models (B^-, \mathbf{p})$ or
  2. $H(\mathbf{p}) = 0$

Hybrid Answer Set Programming (H-ASP)

Stable Model

A **Gelfond-Lifschitz reduct** of P over M and I, $P^{M,I}$:

1. Eliminate all the inconsistent rules
2. If an advancing rule $a \leftarrow B : A, O$ is not eliminated then it is replaced by $a \leftarrow B^{+} : A^{+}, O^{+}$ where O^{+} is the set $\mathbf{p} \in O \cap \left(\widehat{M} \cup \{I\}\right)$ s.t. $M \models (B^{-}, \mathbf{p})$ and $A(\mathbf{p}) \cap \widehat{M} \neq \emptyset$ and $A^{+}(\mathbf{p})$ is defined as $A(\mathbf{p}) \cap \widehat{M}$
3. If a stationary rule $a \leftarrow B_1; B_2 : H, O$ is not eliminated then it is replaced by $a \leftarrow B_1^{+}; B_2^{+} : H|_{O^{+}}, O^{+}$ where O^{+} is the set $(\mathbf{p}_1, \mathbf{p}_2) \in O \cap \left(\widehat{M} \cup \{I\}\right)^2$ s.t. $M \models \left(B_i^{-}, \mathbf{p}_i\right)$ and $H(\mathbf{p}_1, \mathbf{p}_2) = 1$
4. Similarly for a stationary rule $a \leftarrow B : H, O$ that is not eliminated.

M is a **stable model of P with the initial condition I** if

$$T_{P^{M,I},I}(\emptyset) \uparrow \omega = M$$

Hybrid Answer Set Programming (H-ASP)

A H-ASP program of order 1 is a program that consists only of rules of the form

$$a \leftarrow B : G, O$$

Theorem

(informally) Let At be a set of propositional atoms and let S be a set of parameter values. Let $\langle V, E \rangle$ be a computational cDAG over At and S. Then there exists a H-ASP program P of order 1 and an initial condition I for P such that the set of stable models of P with the initial condition I encode $\langle V, E \rangle$.

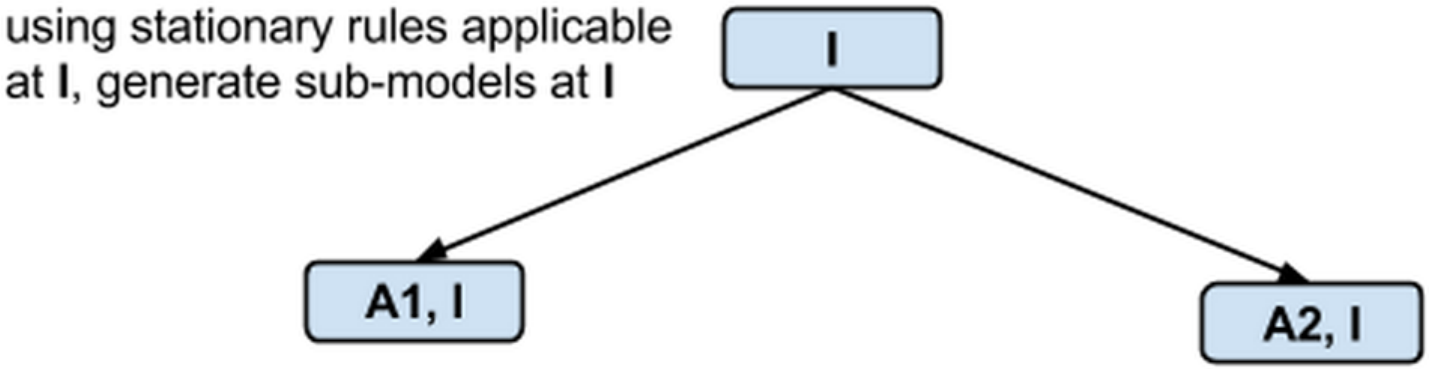
H-ASP PL - H-ASP Python Library

The intuition

start with the initial condition

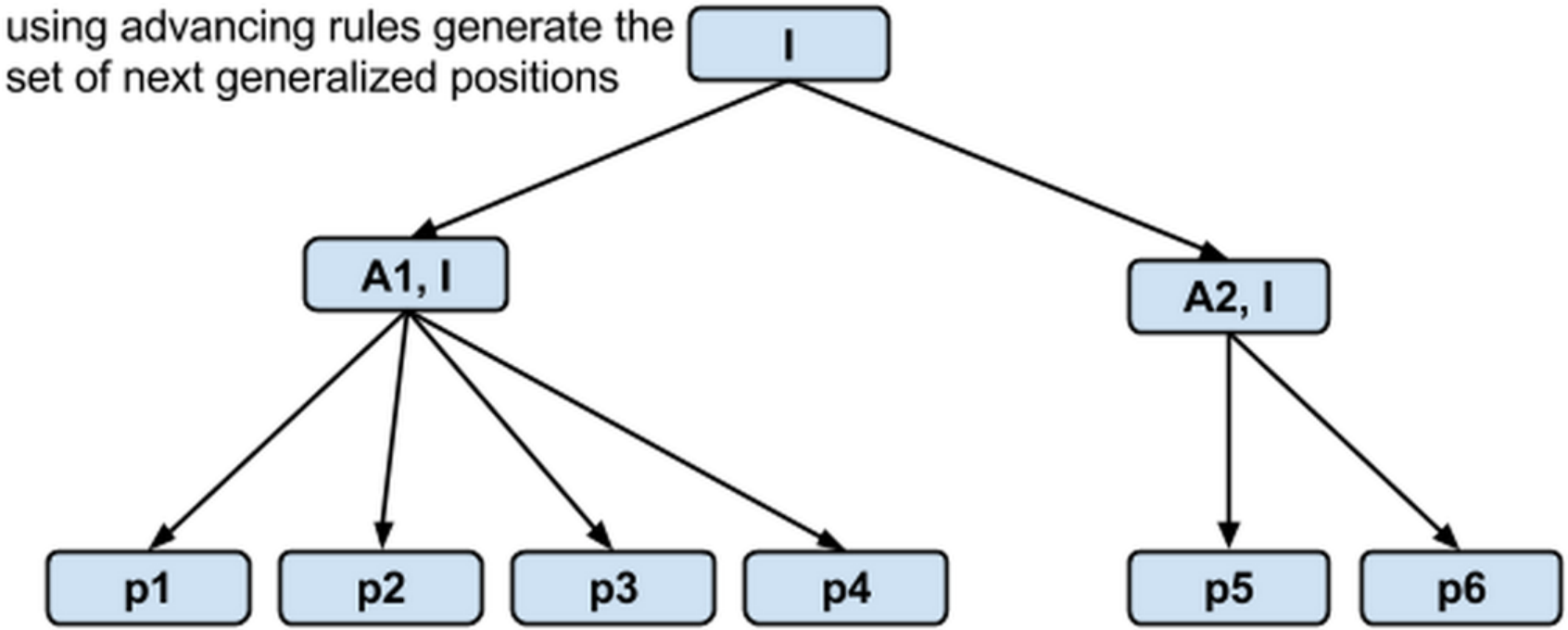
H-ASP PL - H-ASP Python Library

The intuition



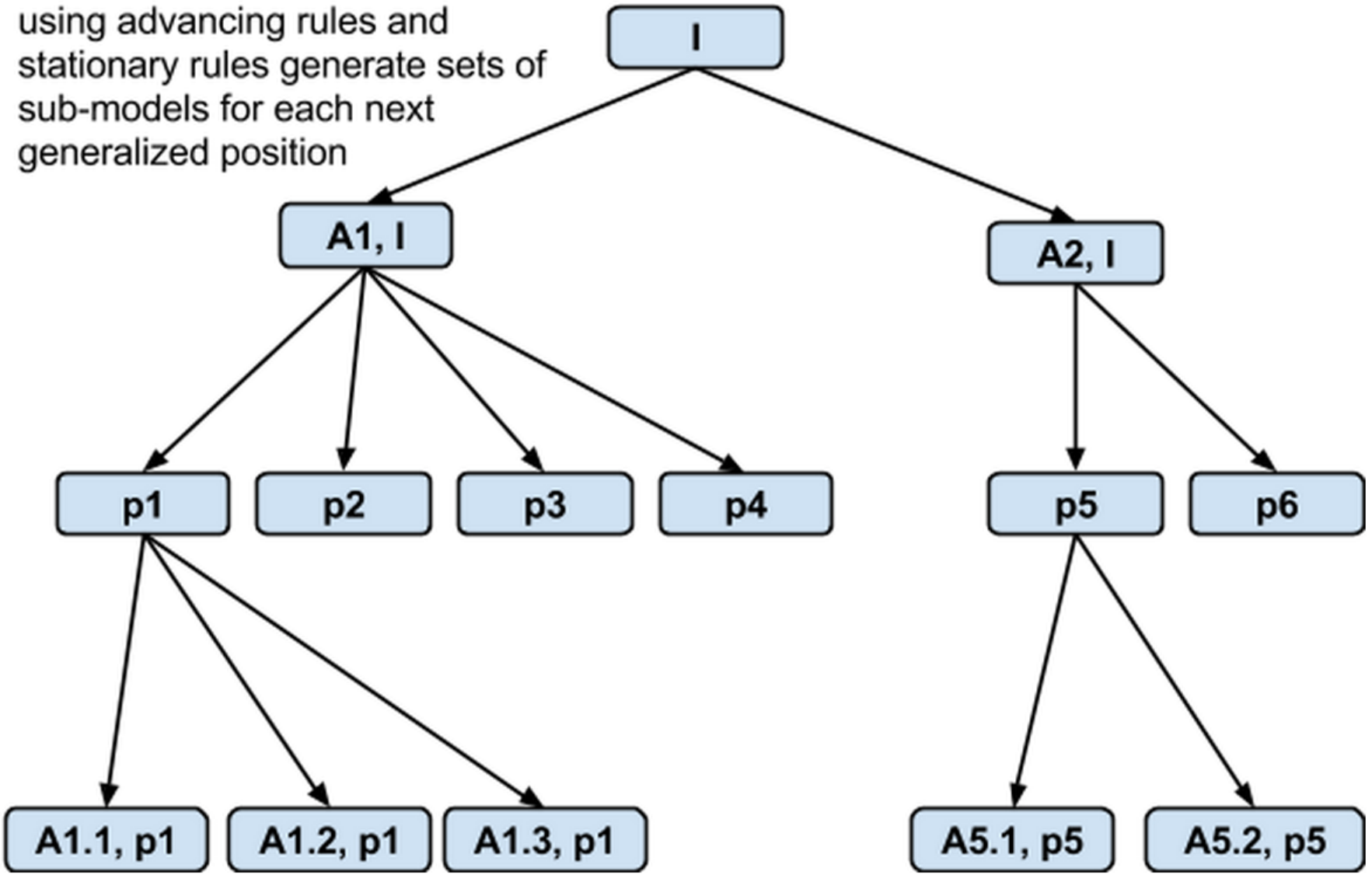
H-ASP PL - H-ASP Python Library

The intuition



H-ASP PL - H-ASP Python Library

The intuition



H-ASP PL - H-ASP Python Library

Features

- Support definition of H-ASP rules

$$a \leftarrow a_1, a_2, not\ b_1;\ a_3, not\ b_2 : A, O$$

into

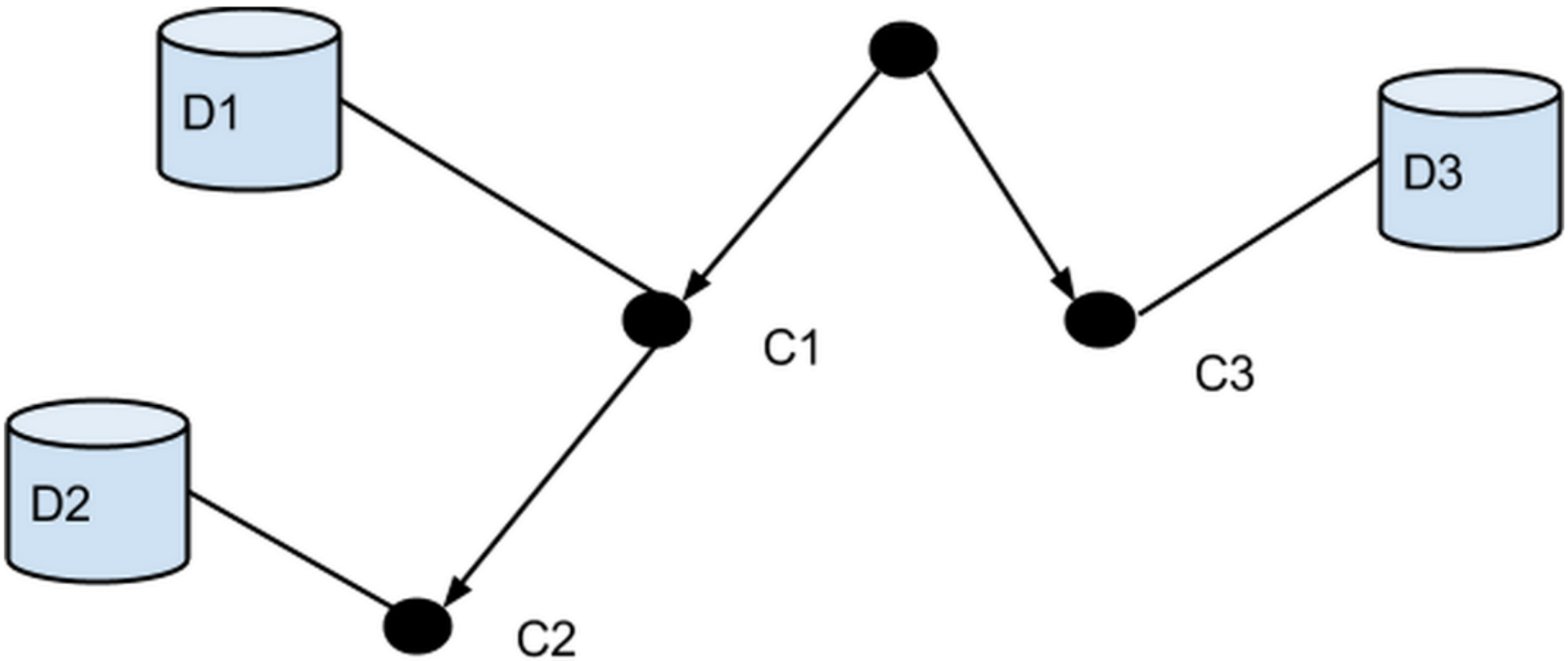
$$DefineRule(\text{"}a\text{"}, \text{"}a1, a2, not\ b1;\ a_3, not\ b2\text{"}, A, O)$$

where A and O are Python functions

- Parameters are named; each advancing algorithm outputs the values for a specified subset of parameters, declared as a signature of the advancing algorithm. The generalized positions are assembled by performing cross products on the outputs of the algorithms.
- For an advancing algorithm A, if $\mathbf{q} \in A(\mathbf{p})$ then $t(\mathbf{q}) = t(\mathbf{p}) + 1$
- For a stationary rule $a \leftarrow B_1; B_2 : H, O$, if the rule is applicable at $(\mathbf{p}_1, \mathbf{p}_2)$ then $\mathbf{p}_2$ is a successor of $\mathbf{p}_1$
- The restrictions can be implemented explicitly by H-ASP rules.

Example: Verifying Advertiser's Readiness

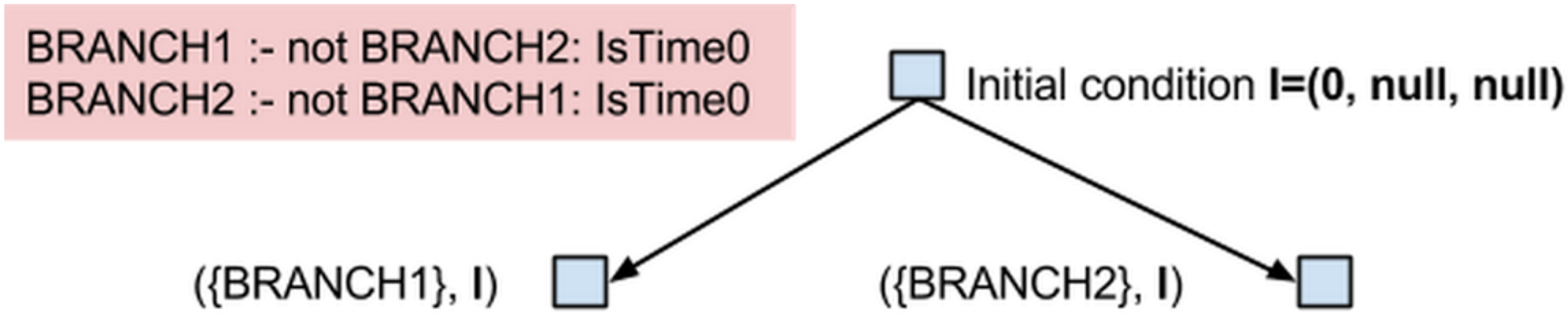
- Constraint $C1$ requires data from $D1$, constraint $C3$ requires data from $D3$
- constraint $C2$ requires data used to check $C1$ and data from $D2$



Example (The Local Algorithm)

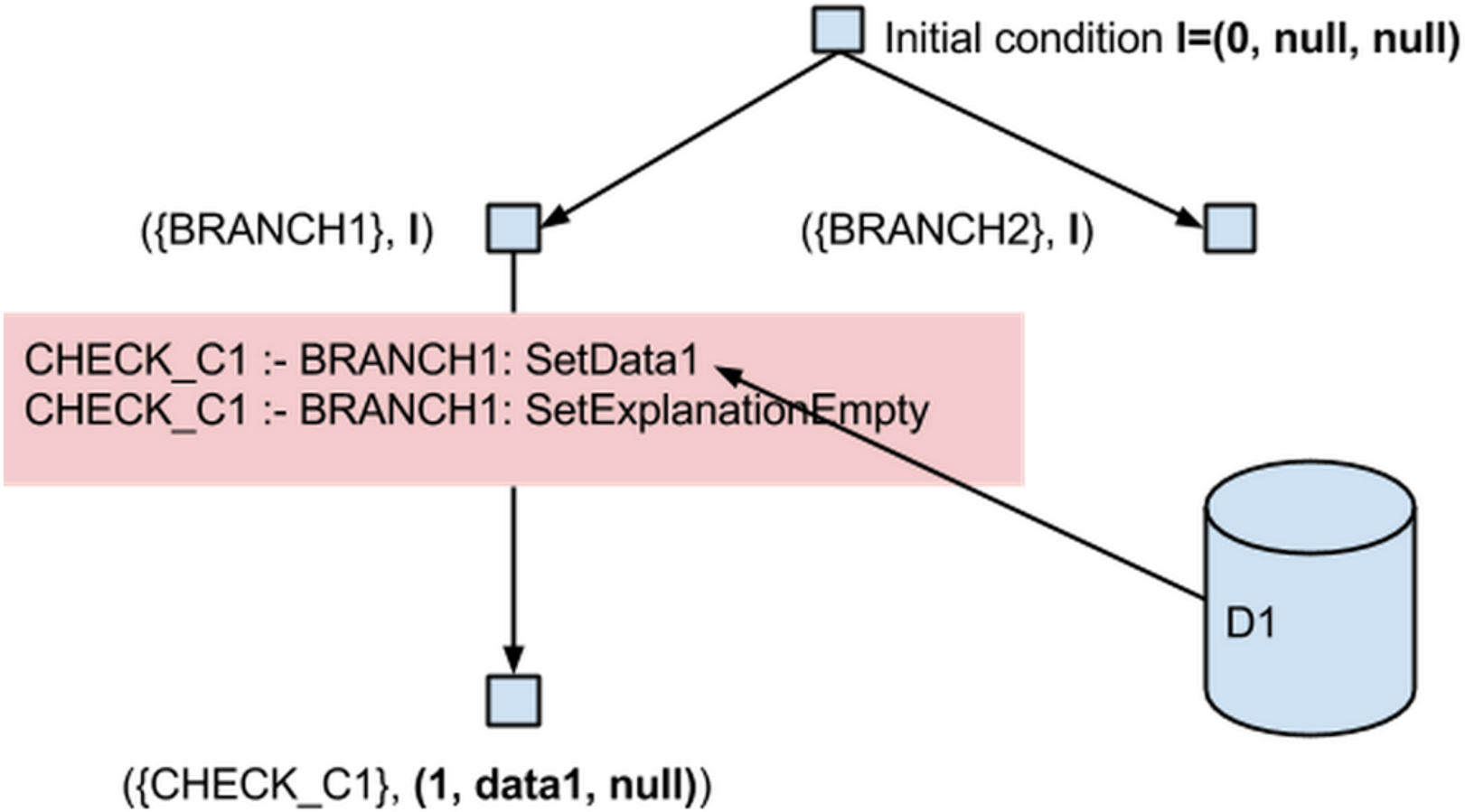
Two named parameters: **DATA** and **EXPLANATION;**

Find stationary rules applicable at I



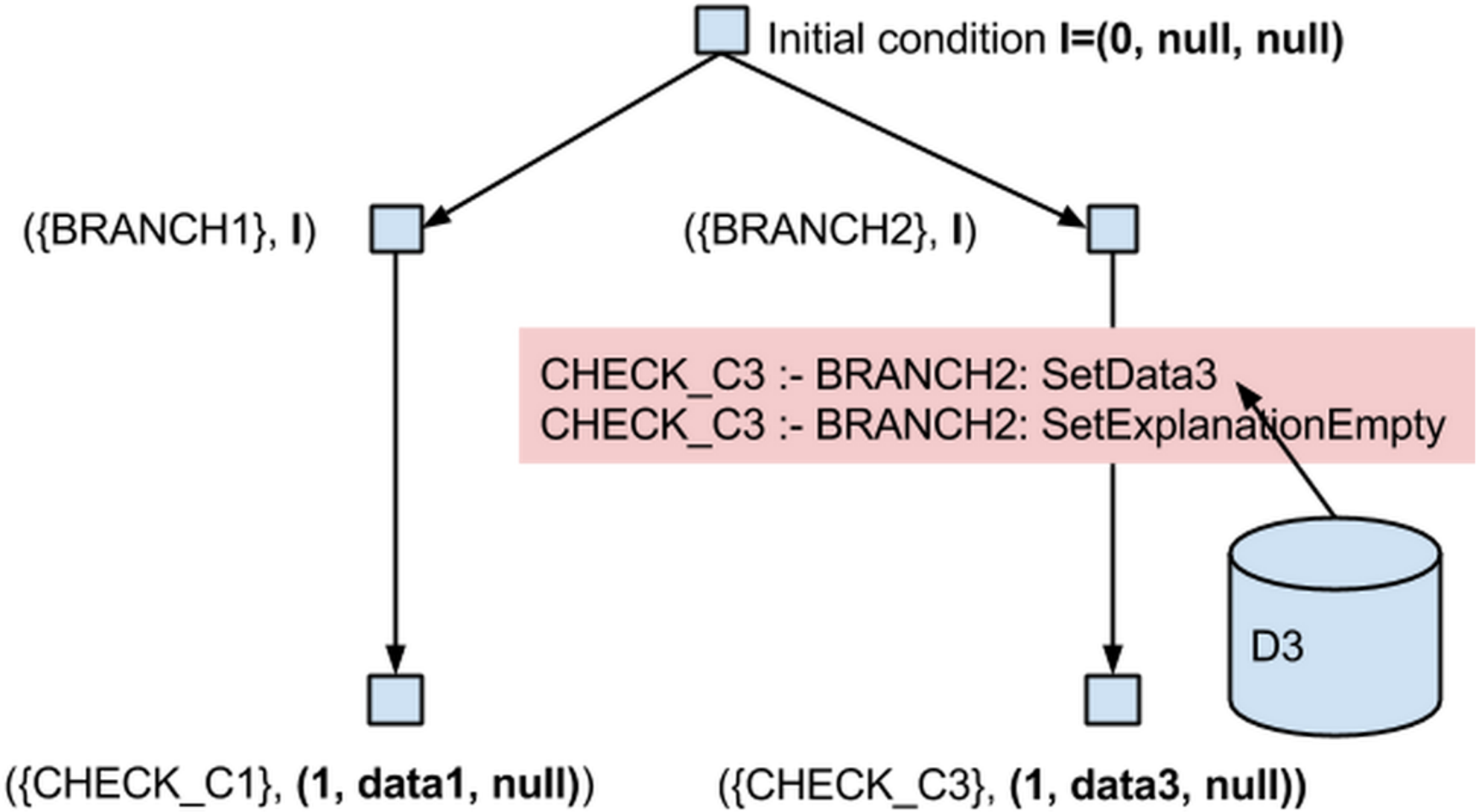
Example (The Local Algorithm)

Find advancing rules applicable at ({BRANCH1}, I)



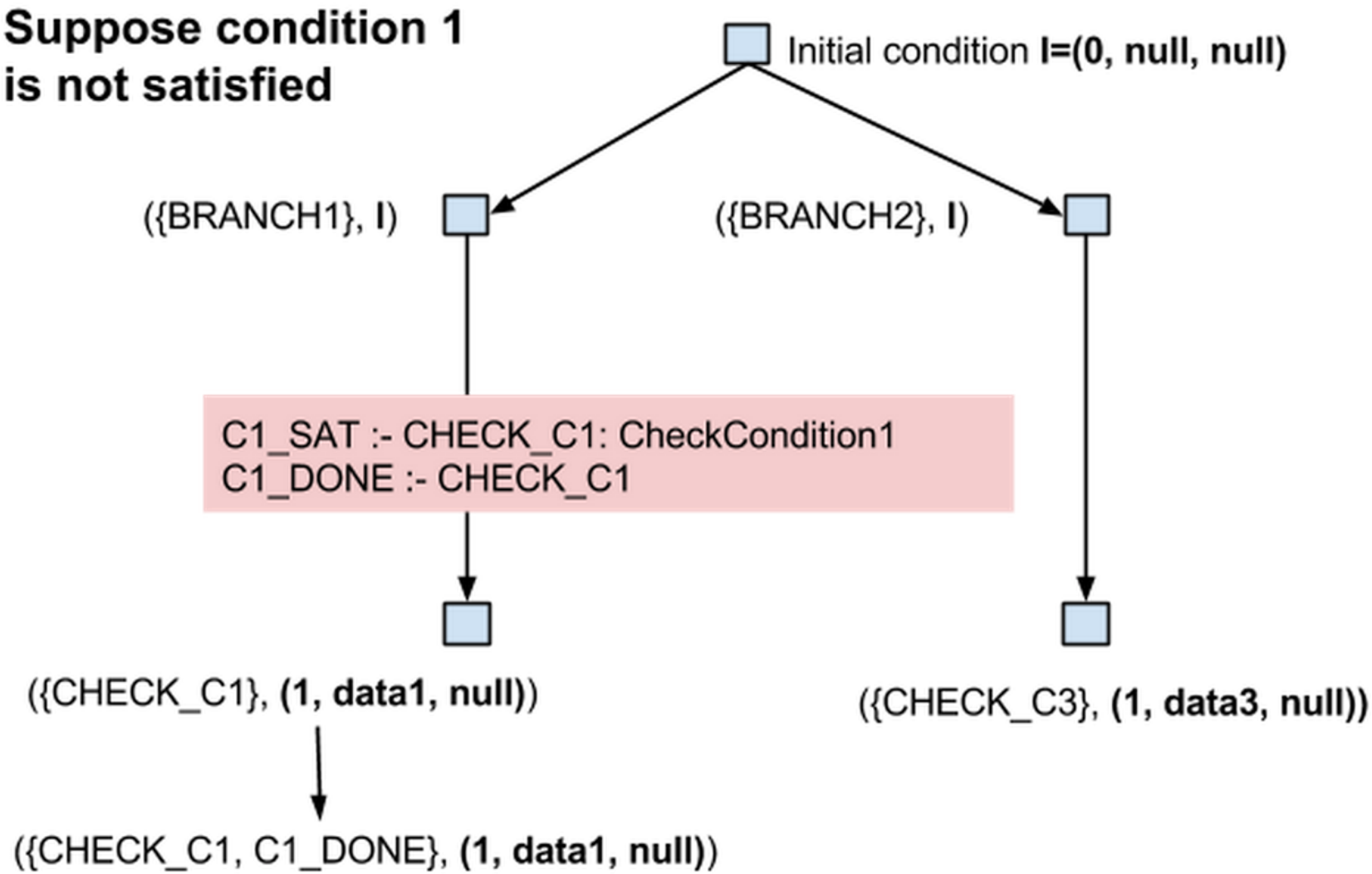
Example (The Local Algorithm)

Find advancing rules applicable at ({BRANCH2}, I)



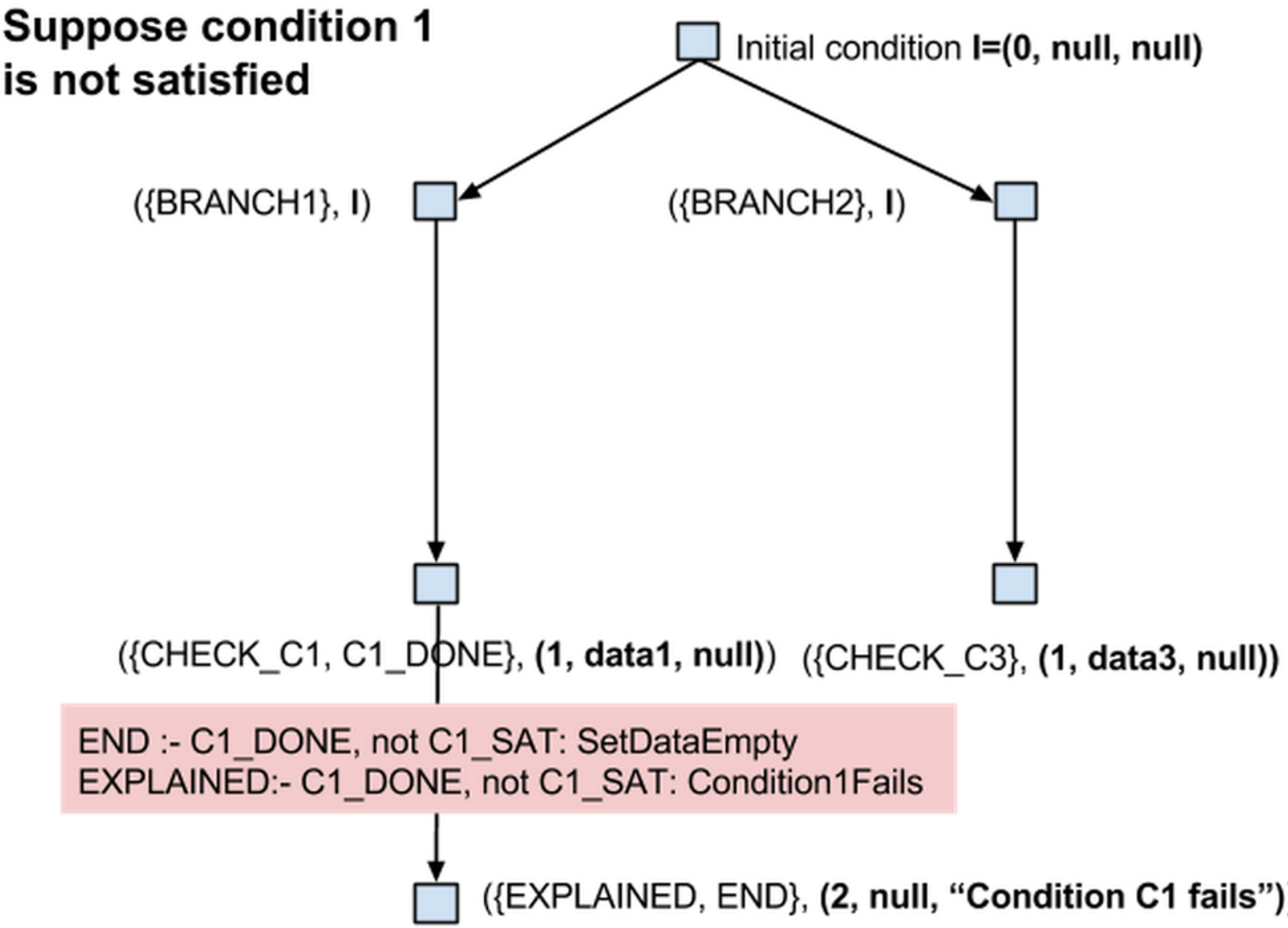
Example (The Local Algorithm)

Find stationary rules applicable at ({CHECK_C1}, (1, data1, null)}



Example (The Local Algorithm)

Find advancing rules applicable at ({CHECK_C1, C1_done}, (1, data1, null)}



The Local Algorithm

Let P be a valid H-ASP PL program and I be an initial condition

1. Use stationary rules applicable at I to derive sub-models at I
2. Repeat for every hybrid state $(A, \mathbf{p})$:
   1. Use advancing rules applicable at $(A, \mathbf{p})$ to generate a set of next candidate states $(D_1, \mathbf{q}_1)$, ..., $(D_n, \mathbf{q}_n)$
   2. For every candidate state $(D_i, \mathbf{q}_i)$ use stationary rules applicable at $((A, \mathbf{p}), (D_i, \mathbf{q}_i))$ and stationary rules applicable at $(D_i, \mathbf{q}_i)$ to generate a set of next hybrid states $(D_{i,1}, \mathbf{q}_i)$, ..., $(D_{i,m_i}, \mathbf{q}_i)$

Semantics of H-ASP PL

- Every valid H-ASP PL program P has an underlying H-ASP program $Tr\left(Tr[PL]\left(P\right)\right)$
- For every valid H-ASP PL program P, $Tr\left(Tr\left[PL\right]\left(P\right)\right)$ has a unique maximal stable model
- A stable model of a valid H-ASP PL program P is defined as a transform of the unique maximal stable model of $Tr\left(Tr\left[PL\right]\left(P\right)\right)$

Theorem

(informal) Every computer representable computational cDAG can be generated by a suitably constructed H-ASP PL program

Theorem

For a valid H-ASP PL program P and an initial condition I of $Tr\left[PL\right]\left(P\right)$, the result of applying the Local Algorithm to P produces the unique maximal stable model of $Tr\left(Tr\left[PL\right]\left(P\right)\right)$ with the (suitably transformed) initial condition $J\left(I\right)$

Generating a diagnosis

- H-ASP PL program W that encodes all the possible ways to automatically whitelist an advertiser (applicable to any advertiser)
- For a specific advertiser c the initial condition $I(c)$ contains the information about the advertiser
- Local Algorithm generates a maximal stable model of W with the initial condition $I(c)$
- The maximal stable model encodes the computational cDAG, with the leaf nodes corresponding to hybrid states containing atom *END*
- A diagnosis exists if all the hybrid states with atom *END* also contain atom *EXPLAINED*
- Diagnosis is created by examining EXPLANATION parameter of all the hybrid states containing atoms *END* and *EXPLAINED*.

Conclusion

- Use of ASP like program allowed rapid update the diagnostic logic: criteria for whitelisting changed many times
- The program was used to diagnose the failures for many advertisers over a time interval of several months
- Reduced the time to diagnose a single failure from 30-60 mins (manual debugging per advertiser) to 1-3 mins

Related Work

- Solving diagnostic problems based on the theory of action language $\mathcal{AL}$ (by Balduccini and Gelfond)
  - (Balduccini and Gelfond) a mathematical model of the dynamic domain
  - Interesting future work; not attempted due to project's time constraints
- Extensions of ASP allowing external data searches: DLV^{DB}, *VI* programs, *GRINGO* grounder, *HEX* programs
- Redl in the PhD thesis notes that HEX programs generalize the above
- HEX programs (Eiter et al.) are an extension of ASP that allow accessing external data sources via external atoms.
- The main differences with H-ASP PL:
  - For H-ASP PL, information processed by algorithm is kept as a separate class of information from logical atoms
  - H-ASP PL has built-in support for producing computational cDAGs
  - H-ASP PL stable models are the unique maximal stable models

Thank you