Concurrent Constraint Programming

The "store": a first-order formula $\sigma$

ask/tell $\varphi$

The demon who answers entailment queries

ask/tell
CCP processes

**ask(φ):** does the current store (σ) entail φ?

**tell(φ):** add φ to the current store.

\[ P_1 || P_2 : \text{run } P_1 \text{ and } P_2 \text{ in parallel.} \]

\[ \text{new } X \text{ in } P : \text{fresh local variable; } \nu X.P. \]

recursive procedures

Underlying (first-order) language and \( \vdash \). The constraint system.

The demon answers entailment queries somehow.

If \( \sigma \not\vdash \phi \) then process suspends.
Examples

More detailed syntax for ask: ask(ϕ) → P.

tell(X = 1)||(ask(X > 0) → (tell(Y = 17)||(ask ...))

Henceforth, I will skip ask and tell.

\[
P_1 : (X > 1)||(Y > 0) → ((X > 2)||(Y > 1) → (X = 17)))
 \]
\[
P_2 : (X > 1) → [(Y > 0)||(X > 2) → (Y = 2)]
\]

When \( P_1 \) and \( P_2 \) are run in parallel they will engage in a dialogue.

Ask is a synchronizer: perhaps it should have been called await.
Constraint system examples

Kahn-McQueen style dataflow: Define a language with function symbols for *cons, first* and *rest* and appropriate predicate symbols and entailment relations to model streams.

CCP is an **asynchronous** programming paradigm.

Herbrand: terms in a first-order language with equality. Constraints are equality statements or their conjunctions. Example entailment:

\[ f(X,Y) = f(Z,g(U,V)) \vdash (X = Z) \land (Y = g(U,V)) \]

Rational intervals.
The partial order of the denotational semantics is now something with which one can program.

Programs update the store, but only in a monotonic way.

Demons answer entailment queries: the store can be thought of as the theory that it generates.

The collection of possible stores can be ordered by inclusion of the theories they define.

This collection forms a complete algebraic lattice if we make some assumptions about constraint systems.
Probabalistic CCP

New ingredient: \textbf{choose } \textit{X from Dom in } \textit{P}

\textit{X}: local variable, scope is \textit{P}

\textbf{Dom} is a finite set

\begin{itemize}
  \item Random variables are hidden
  \item Each random variable has its own independent probability distribution.
\end{itemize}
Basic Example

choose $X$ from $\{0, 1, 2, 3\}$ in

$$[\text{ask}((X = 0) \lor (X = 1)) \rightarrow \text{tell}(a)] \lor [\text{ask}(X = 2) \rightarrow \text{tell}(b)]$$

Produces $a$ with probability 0.5 and $b$ with 0.25.

and $\text{true}$ with probability 0.25.
Constraints and conditioning

choose $X$ from $\{0, 1, 2, 3\}$ in

\[
\text{tell}(X \leq 2) \ || \ [\text{ask}((X = 0) \lor (X = 1)) \rightarrow \text{tell}(a)] \ || \\
[\text{ask}(X = 2) \rightarrow \text{tell}(b)]
\]

Produces $a$ with probability 0.5 and $b$ with probability 0.25, however, it cannot produce true because of the constraint on $X$.

Inconsistent stores are discarded and the probabilities are renormalized.

Probability of $a = \frac{2}{3}$ and probability of $b = \frac{1}{3}$.

The semantics gives the probabilities conditioned on obtaining a consistent store.
Notational change

I will stop writing ask, tell and $||$.

$$(X \leq 2), [((X = 0) \lor (X = 1)) \rightarrow a], [(X = 2) \rightarrow b]$$

instead of

$$\text{tell}(X \leq 2) \parallel [\text{ask}((X = 0) \lor (X = 1)) \rightarrow \text{tell}(a)] \parallel [\text{ask}(X = 2) \rightarrow \text{tell}(b)]$$
Independence of `choose`

choose \(X\) from \(\{0,1\}\) in \([X = Z]\),

choose \(Y\) from \(\{0,1\}\) in \([Z = 1 \rightarrow (Y = 1)]\)

Four possible execution paths but one is inconsistent with the constraints.

We get the following distribution on the visible variable \(Z\):
\(Z = 0 (\text{prob } = \frac{2}{3}), \quad Z = 1 (\text{prob } = \frac{1}{3}).\)

We can get any distribution with rational probabilities on a finite set this way.

Derived combinator:
choose \(X\) from \(\text{Dom}\) with \(f\) in \(P\).
What about recursion?

We get distributions on continuous spaces.

\[ U(l, u, z) :: z \in [l, u], \]

choose \( X \) from \( \{0, 1\} \) in

\[
(X = 0) \rightarrow U(l, (u + l)/2, z),
\]

\[
(X = 1) \rightarrow U((u + l)/2, u, z)
\]

Defines the uniform distribution on \([0, 1]\).

Actually it defines a measure on the space of binary sequences but this is Borel isomorphic to \([0, 1]\).
What about conditioning in the presence of recursion?

\[ U(0, 1, z), (z = 0) \]

Intuitively probability of \((z = 0) = 1\), since \(z = 0\) is the only possible output.

However, \(U\) defines a distribution which assigns probability 0 to \((z = 0)\).

When we try to normalize we get nonsense.
Use the “domain” Luke!

Unwind and consider “finite approximations” of the recursive program.

The approximation $U_n$ yields $z = 0$ with positive probability, so $U_n(0, 1, z), (z = 0)$ gives $(z = 0)$ with probability 1.

In the limit we get $(z = 0)$ with probability 1.
Integration example

\[ f : [a, b] \rightarrow [c, d] \text{ a Riemann-integrable function.} \]

\[ P :: U(a, b, X), U(c, d, Y), \]
\[ ((Y < f(X) \rightarrow A), (Y > f(X) \rightarrow B) \]

Let \( R = (b - a) \times (d - c). \)

Our semantics gives: probability of \( A = \frac{1}{R} \times \int_a^b (f - c). \)

More precisely, we get a lower Riemann sum for \( A \) and an upper Riemann sum for \( B \).
Not the Cantor set

\[ NC(l, u, z) :: NC(l, (u + 2l)/3, z), \]
\[ NC((2u + l)/3, u, z), \]
\[ ((u + 2l)/3 < z < (2u + l)/3) \rightarrow \text{NotCantor}. \]

This program produces the token \text{NotCantor} if \( z \) is not in the Cantor set.

If we run \( U(0, 1, X), NC(0, 1, X) \) we get \text{NotCantor} with probability 1.

Our semantics successfully produces the right answers for these examples.
Conditioning on a set of measure 0

Use *computational* approximation to provide answers, where Radon and Nikodym would give up.

```plaintext
new X, Y in [U(0, 1, X), U(0, 1, Y), X = Y]
```

Intuition: uniform measure on the diagonal.

Probability theory: undefined.

Our semantics: gives the intuitive answer.

Note: it does **not** depend on the two recursions being unwound at the same “rate”.
The last example

\[ V(l, u, X) : \quad z \in [l, u], \]

**choose** \( X \) **from** \( \{0, 1\} \) **with** \( \{1/3, 2/3\} \) **in** \[
(X = 0) \rightarrow V(l, (2l + u)/3, z), \\
(X = 1) \rightarrow V((2l + u)/3, u, z)\]

Gives the uniform distribution on \([l, u]\).

Now consider \( U(0, 1, X), V(0, 1, Y), (X = Y) \).

Does not give the uniform distribution on the diagonal.

\[ \Pr((0, 0.1)) < \Pr((0.9, 1.0)). \]

Get a “fractal-like” distribution on the diagonal.
Conclusions

Subtle interplay between conditioning and recursion.

Markov kernels are probabilistic relations.

Limit theorems in measure theory crucial for probabilistic semantics.

New directions: probability at higher types, measures on top of domains, domains of measures.

Got to work with the machine learning people who are the "market" and the source of many important applications.