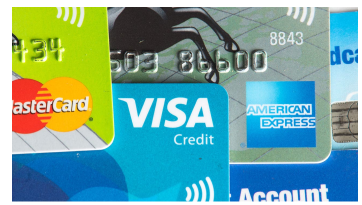
Insurance Portfolio Management

Michael Genesereth
Computer Science Department
Stanford University

The presentation begins with a brief introduction to insurance portfolio management. This is followed by a demonstration of a computer system that supports insurance portfolio management. The presentation then concludes with a look at the underlying technology used in the demonstration.

Insurance Policies



We often think of insurance policies as being partitioned into distinct areas – home insurance, auto insurance, travel insurance, credit card insurance, and so forth; and we frequently buy different policies to provide coverage in these different areas. In reality, things are more complicated, with policies in different areas often providing overlapping coverage. For example, rental car damage may be covered by a policy purchased from a rental car company, a personal auto insurance policy, a credit card, a travel insurance policy, and even, in some cases, a home insurance policy. If we are unaware of these overlaps, we can end up paying more for insurance than we need; what's worse, there can be gaps between policies of which we are unaware

Insurance Portfolio Management

Insurance portfolio management (IPM) is the process of
managing *overlapping insurance policies*
with an eye to *minimizing insurance costs*
while *ensuring adequate coverage*.

Insurance portfolio management (IPM) is the process of managing multiple, potentially overlapping insurance policies, with an eye to minimizing insurance costs while ensuring adequate coverage. By studying the contracts associated with insurance policies, we can avoid overlaps and ensure there are no gaps in coverage.

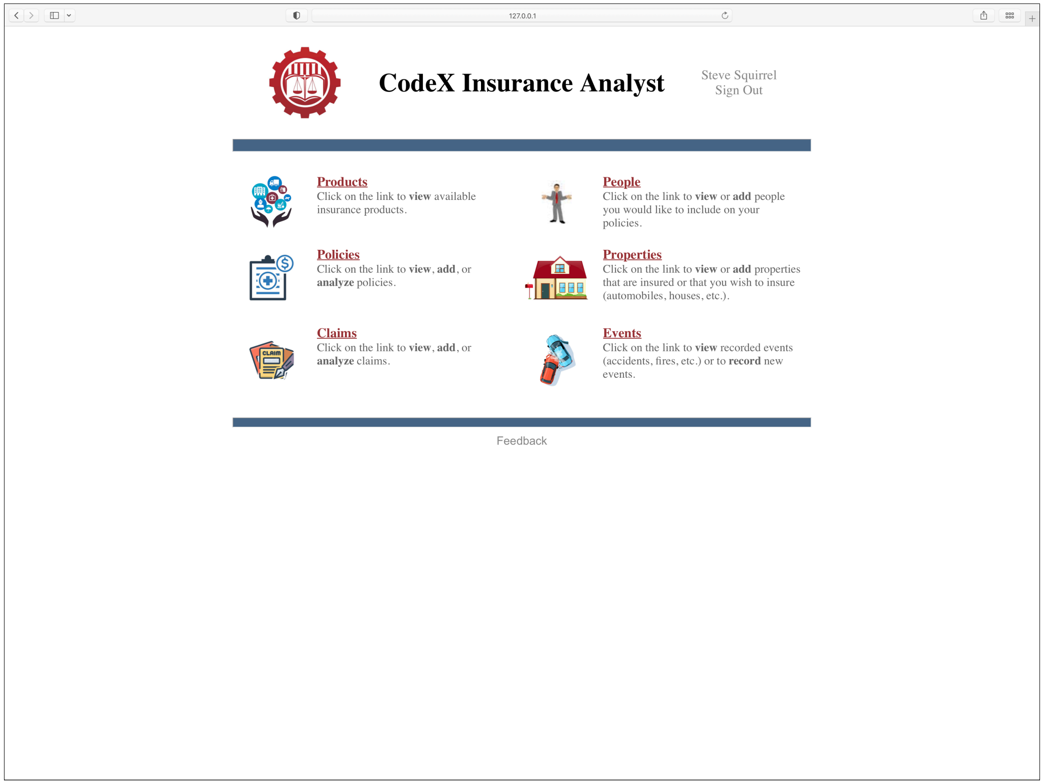
Example

For example, while renting a car, we may realize that we do not need to purchase collision insurance from Hertz because we are already covered if we use our Visa credit card. At the same time, we may realize that we need to purchase additional insurance for travel in Ireland, since our credit card insurance does not apply there.

Complexity of Insurance Policies

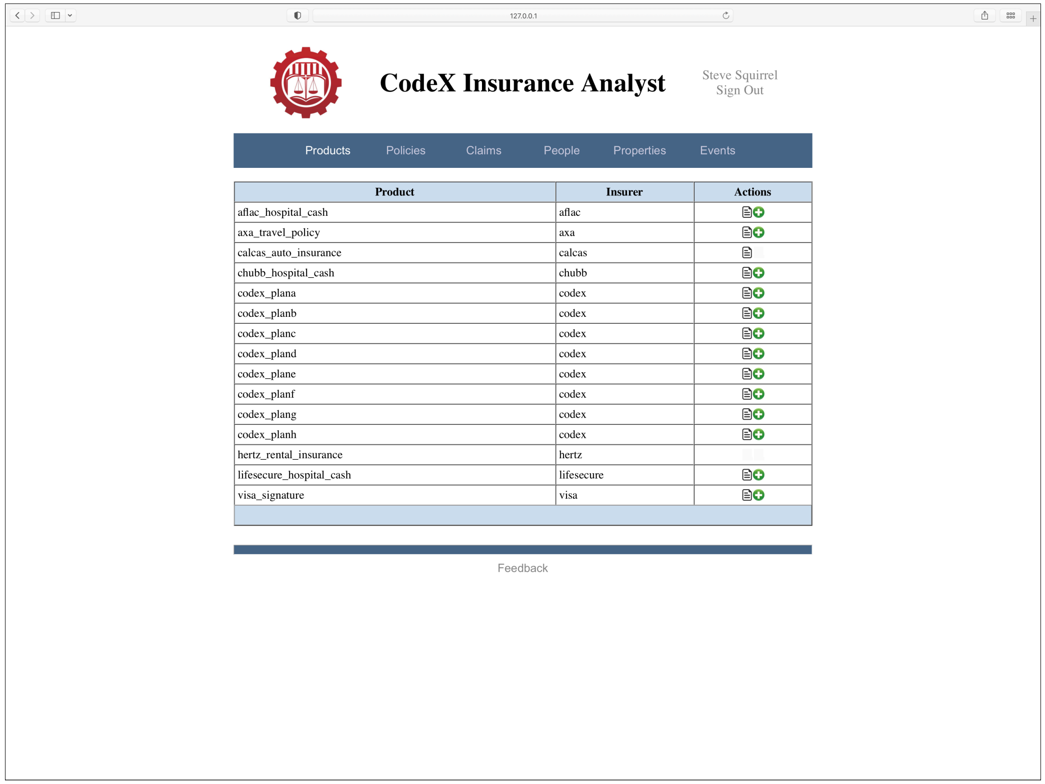
Policies are complex, incomplete, inconsistent.

The problem is that insurance portfolio management is not easy. Insurees usually do not have the time or patience to compare policies from multiple insurance providers; and, even if they have the time, they often do not have the legal background needed to understand the complex legal wording of the lengthy, 100-page contracts associated with those policies.



In order to deal with this problem, we have been investigating ways to automate the process of insurance portfolio management. The current manifestation of this work a web-based application called the CodeX Insurance Analyst (CIA). CIA incorporates data and policies from multiple insurance companies, and it provides users with analysis across these policies. We are currently focussed on individuals. In the future, we will provide similar capabilities for businesses, insurance companies, and insurance regulators.

The first page one sees after signing in is the splash page for the system. The tabs here indicate the principals types of entities represented in the system - insurance products, policies, claims, people, properties (such as vehicles of houses), and events (such as accidents and hospitalizations).



127.0.0.1

CodeX Insurance Analyst

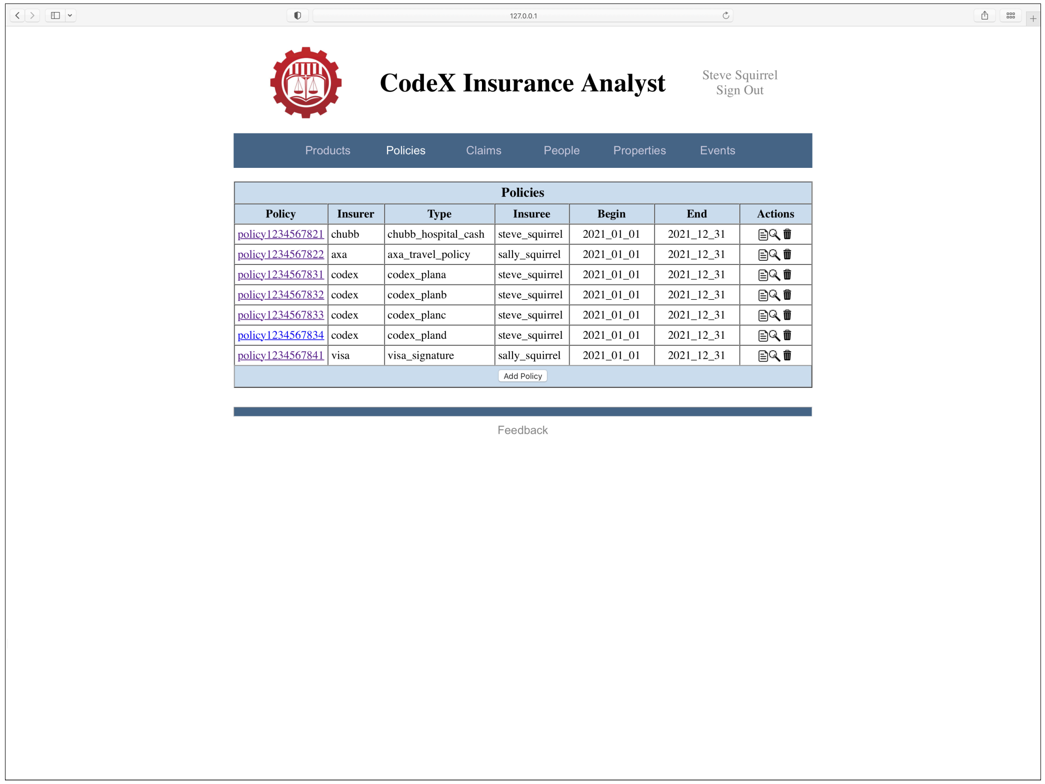
Steve Squirrel
Sign Out

Products Policies Claims People Properties Events

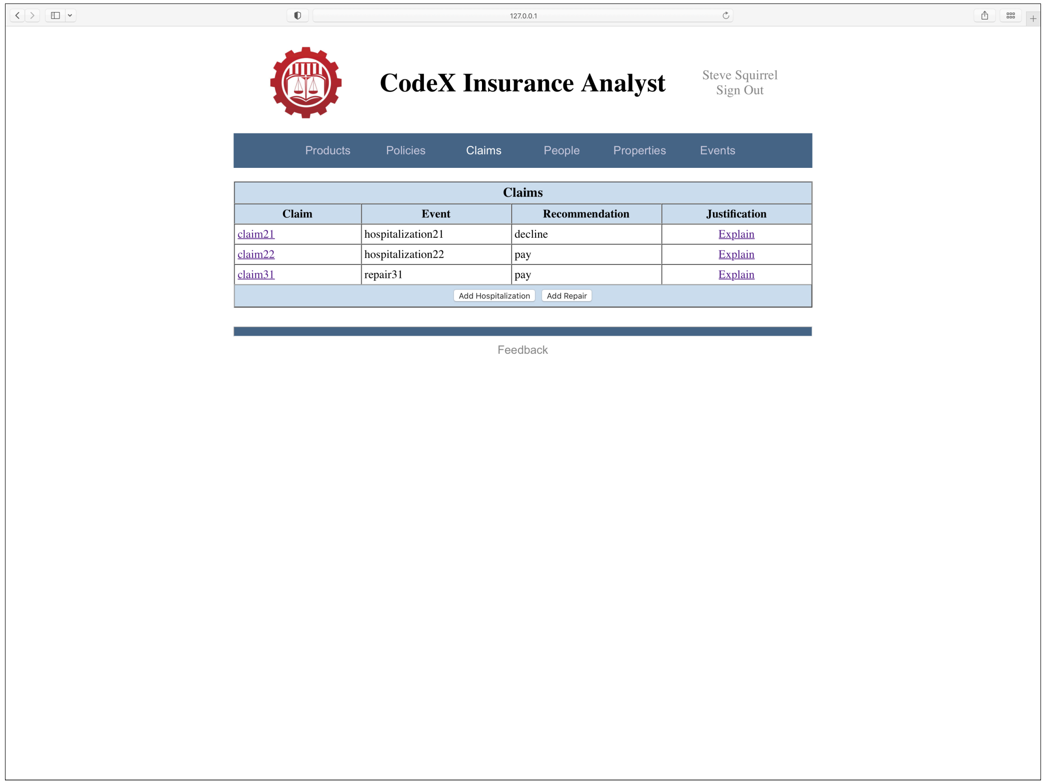
Product	Insurer	Actions
aflac_hospital_cash	aflac	
axa_travel_policy	axa	
calcas_auto_insurance	calcas	
chubb_hospital_cash	chubb	
codex_plana	codex	
codex_planb	codex	
codex_planc	codex	
codex_pland	codex	
codex_plane	codex	
codex_planf	codex	
codex_plang	codex	
codex_planh	codex	
hertz_rental_insurance	hertz	
lifesecure_hospital_cash	lifesecure	
visa_signature	visa	

Feedback

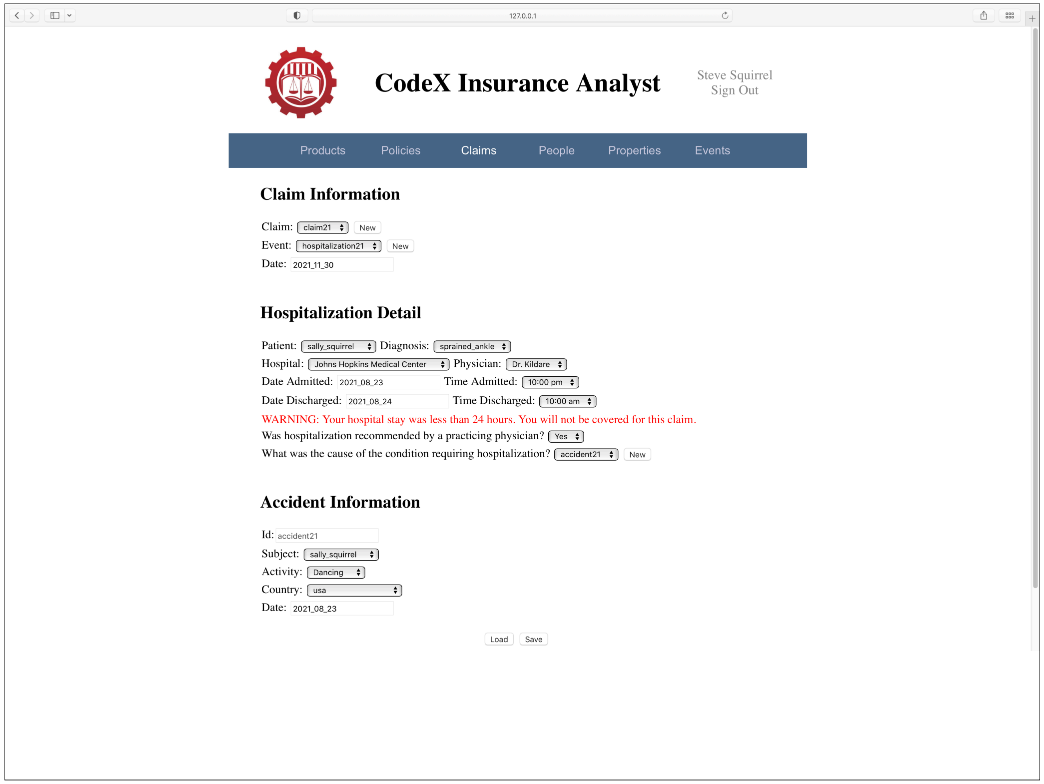
The products page contains a list of insurance products available in the demonstration system. Clicking on the text icon shows the policy. It is possible to add a policy for analysis by clicking the + button. More on that later.



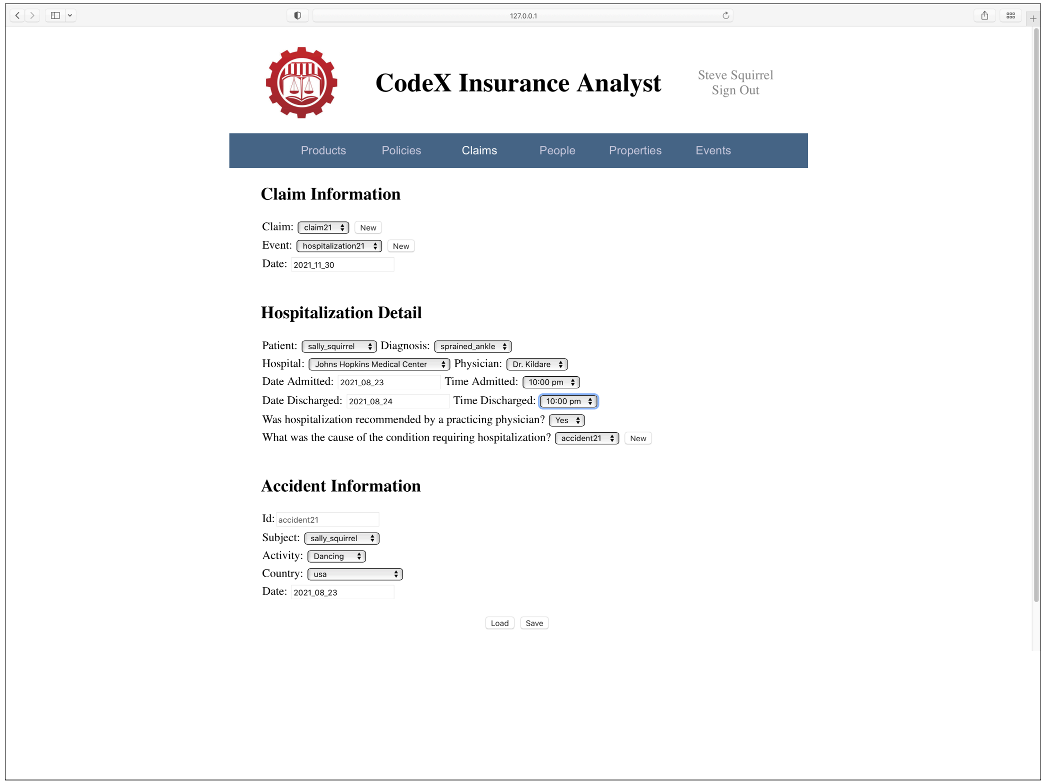
The Policies page shows policies the user has selected for analysis. A policy is essentially a specific insurance product, specialized to a particular person, for a specific range of dates, with desired riders, and so forth. Clicking on a text icon to see product description. Clicking on a magnifier requests a comparative analysis of that policy. More on this later. Clicking the trash can deletes a policy.



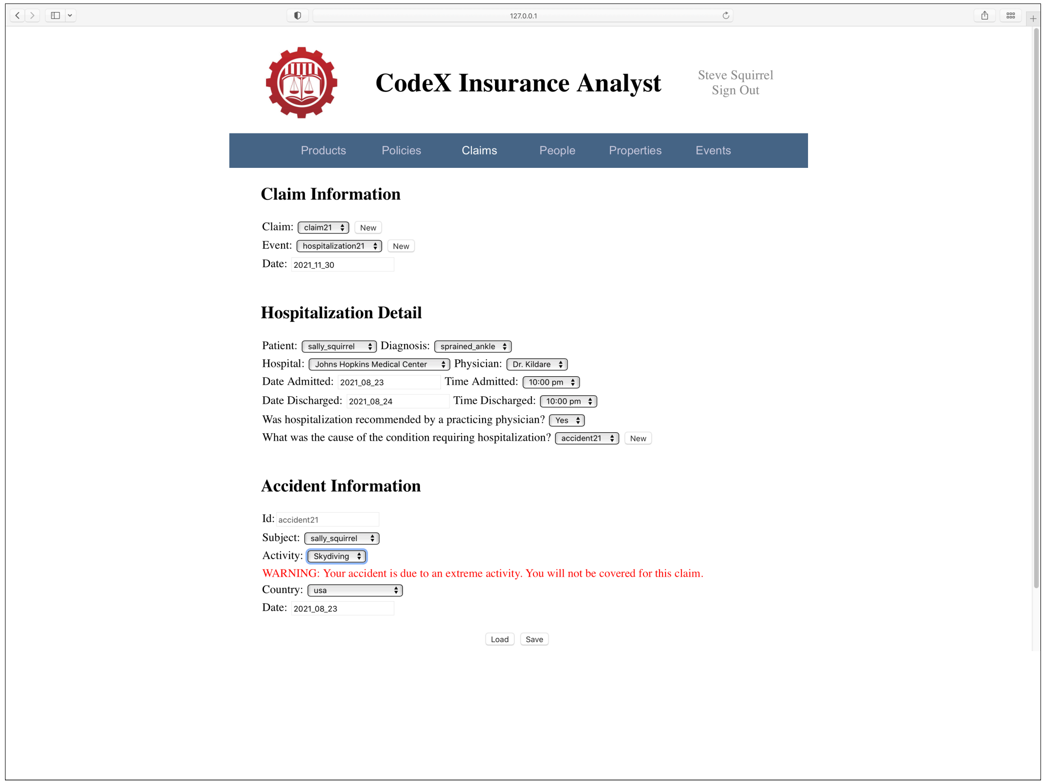
The Claims page shows claims the user would like to analyze. In Steve's case, there are two hospitalization claims and one vehicle repair claim already specified. According to the policies Steve has selected, one of the hospitalization claims should be declined, while the other should be paid. Clicking the explanation button next to a claim gives an explanation of the listed recommendation. Additional claims can be added by clicking one of the Add buttons.



In order to see how claim analysis works, let's look at one of these claims. Clicking on the identifier for a claim brings up a form with details about that claim. The panel at the top provides overall information about the claim. The panel in the middle shows details of the relevant hospitalization. In this case, Steve's daughter Sally was treated for a sprained ankle at Johns Hopkins. At the bottom, is information about the event that led to the injury, in this case a mishap on the dance floor. In this case, the claim is denied because Sally's stay at the hospital was too short - just 12 hours.

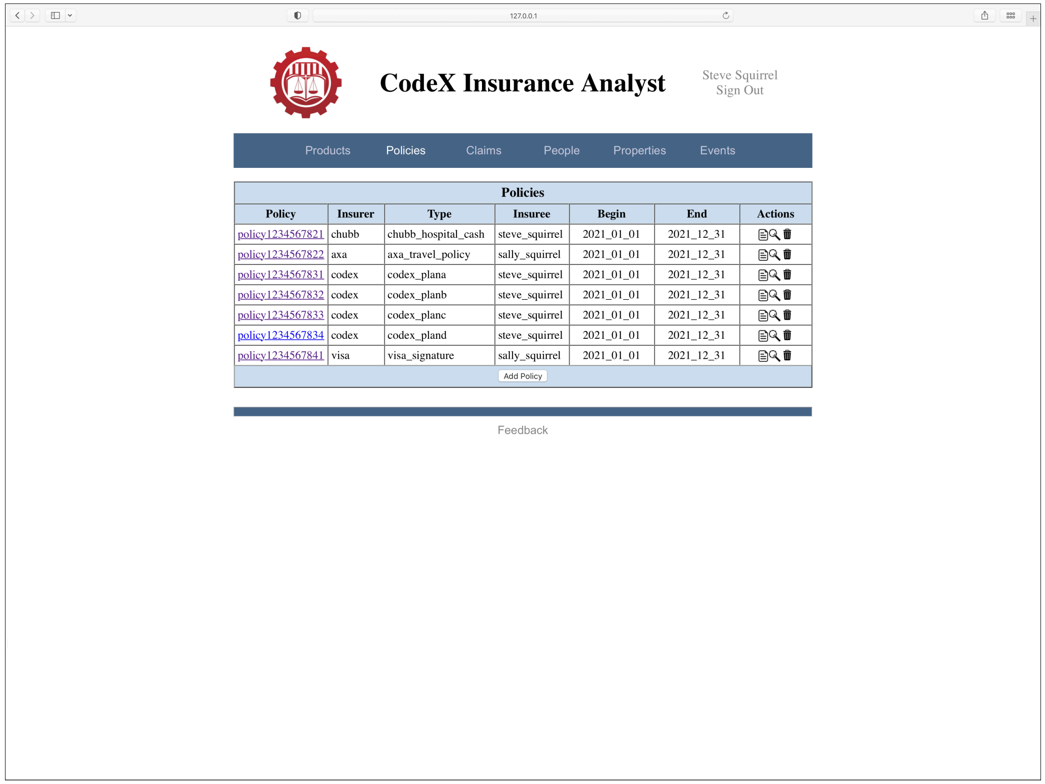


If we adjust the length of the stay, we notice that the claim is acceptable.



However, if Sally's sprained ankle was the result of skydiving rather than dancing, the claim would once again be declined since that is a hazardous activity that is explicitly excluded.

The main feature to note here is the transparency provided by the worksheet. The rules defining the terms and conditions of the customer's policies are embedded in the worksheet and that allows the worksheet to give the claimant immediate feedback on the claim would be honored. Changing various values changes the prediction. Moreover, the worksheet is able to provide explanations for its conclusions so the claimant understands the reasons for its conclusions.



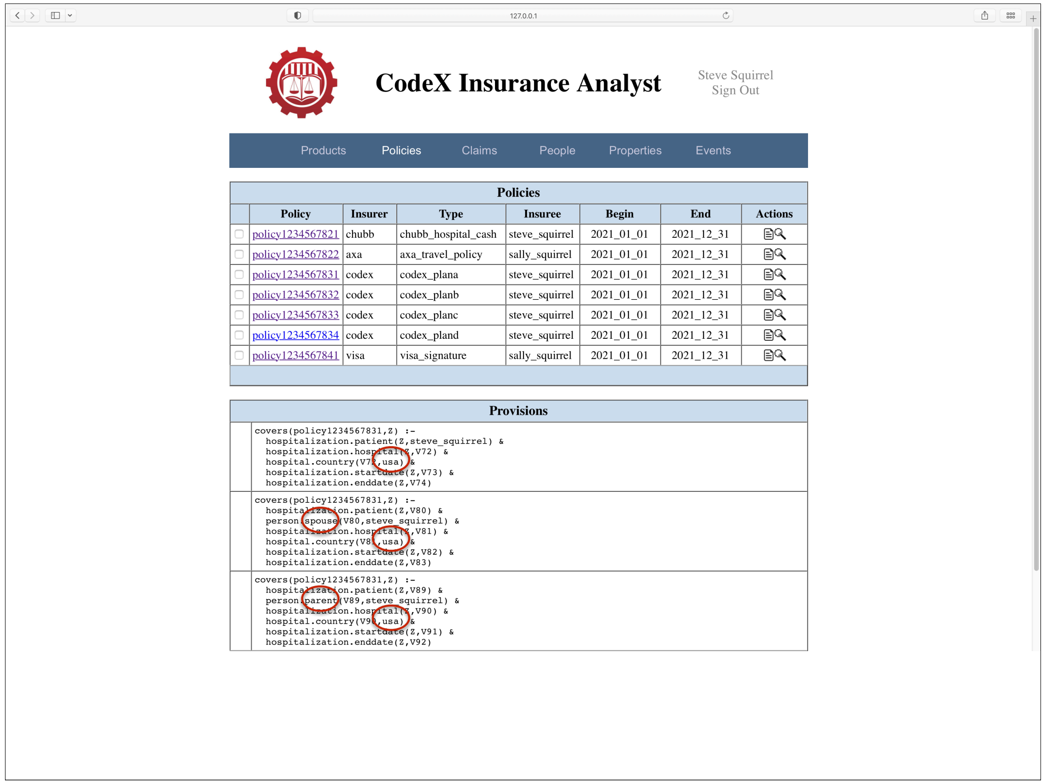
Policies

Policy	Insurer	Type	Insuree	Begin	End	Actions
policy1234567821	chubb	chubb_hospital_cash	steve_squirrel	2021_01_01	2021_12_31	
policy1234567822	axa	axa_travel_policy	sally_squirrel	2021_01_01	2021_12_31	
policy1234567831	codex	codex_plana	steve_squirrel	2021_01_01	2021_12_31	
policy1234567832	codex	codex_planb	steve_squirrel	2021_01_01	2021_12_31	
policy1234567833	codex	codex_planc	steve_squirrel	2021_01_01	2021_12_31	
policy1234567834	codex	codex_pland	steve_squirrel	2021_01_01	2021_12_31	
policy1234567841	visa	visa_signature	sally_squirrel	2021_01_01	2021_12_31	

Add Policy

Feedback

Back to the policies page.



Policies

	Policy	Insurer	Type	Insuree	Begin	End	Actions
	policy1234567821	chubb	chubb_hospital_cash	steve_squirrel	2021_01_01	2021_12_31	
	policy1234567822	axa	axa_travel_policy	sally_squirrel	2021_01_01	2021_12_31	
	policy1234567831	codex	codex_plana	steve_squirrel	2021_01_01	2021_12_31	
	policy1234567832	codex	codex_planb	steve_squirrel	2021_01_01	2021_12_31	
	policy1234567833	codex	codex_planc	steve_squirrel	2021_01_01	2021_12_31	
	policy1234567834	codex	codex_pland	steve_squirrel	2021_01_01	2021_12_31	
	policy1234567841	visa	visa_signature	sally_squirrel	2021_01_01	2021_12_31	

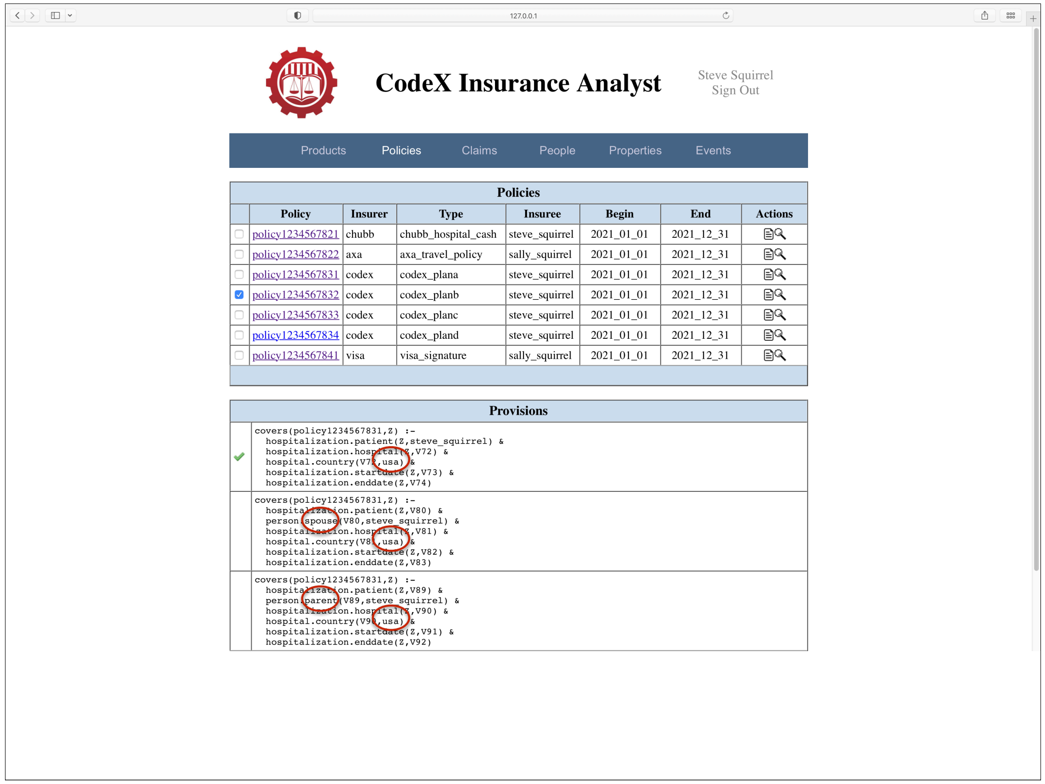
Provisions

```
covers(policy1234567831,Z) :-
  hospitalization.patient(Z,steve_squirrel) &
  hospitalization.hospital(Z,V72) &
  hospital.country(V72,usa) &
  hospitalization.startdate(Z,V73) &
  hospitalization.enddate(Z,V74)
```

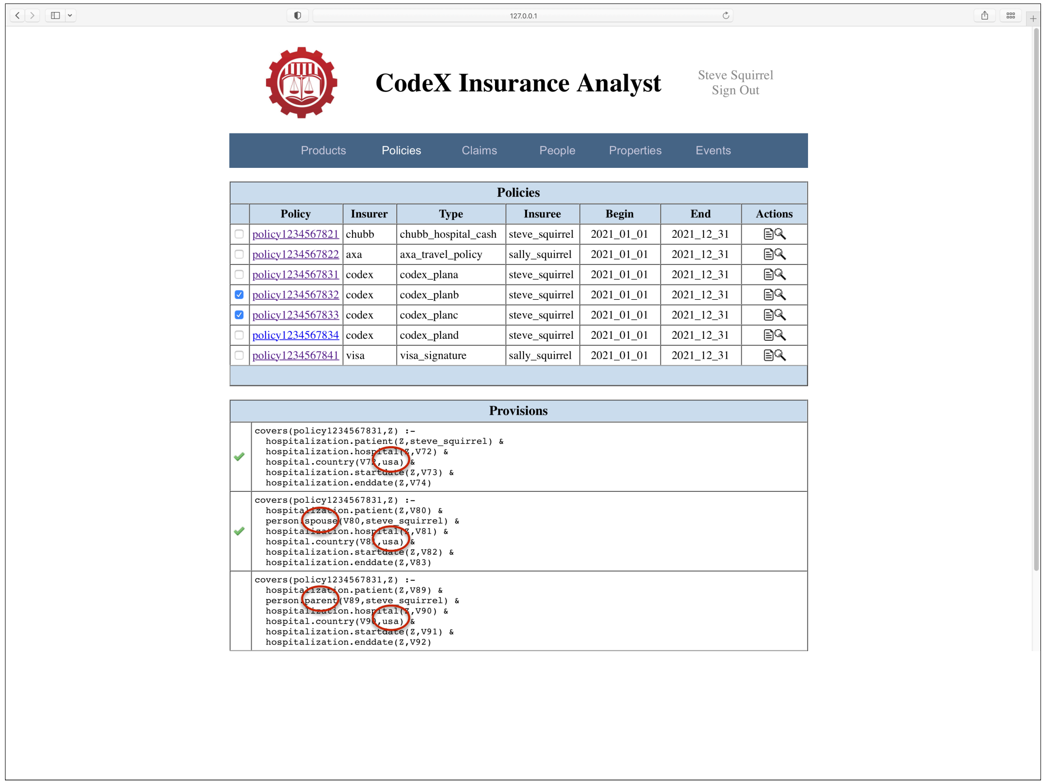
```
covers(policy1234567831,Z) :-
  hospitalization.patient(Z,V80) &
  person.spouse(V80,steve_squirrel) &
  hospitalization.hospital(Z,V81) &
  hospital.country(V81,usa) &
  hospitalization.startdate(Z,V82) &
  hospitalization.enddate(Z,V83)
```

```
covers(policy1234567831,Z) :-
  hospitalization.patient(Z,V89) &
  person.parent(V89,steve_squirrel) &
  hospitalization.hospital(Z,V90) &
  hospital.country(V90,usa) &
  hospitalization.startdate(Z,V91) &
  hospitalization.enddate(Z,V92)
```

Clicking on the magnifier next to codex_plana policy shows the provisions of the policy - steve, spouse, and kids covered for hospitalizations in a US hospital.



Recall that plan b covers steve in US hospitals. Checking the planb box leads to a checkmark next to the first provision. On other words, plan b covers the first provision.



CodeX Insurance Analyst

Steve Squirrel
Sign Out

Products | Policies | Claims | People | Properties | Events

Policies

	Policy	Insurer	Type	Insuree	Begin	End	Actions
☐	policy1234567821	chubb	chubb_hospital_cash	steve_squirrel	2021_01_01	2021_12_31	
☐	policy1234567822	axa	axa_travel_policy	sally_squirrel	2021_01_01	2021_12_31	
☐	policy1234567831	codex	codex_plana	steve_squirrel	2021_01_01	2021_12_31	
☑	policy1234567832	codex	codex_planb	steve_squirrel	2021_01_01	2021_12_31	
☑	policy1234567833	codex	codex_planc	steve_squirrel	2021_01_01	2021_12_31	
☐	policy1234567834	codex	codex_pland	steve_squirrel	2021_01_01	2021_12_31	
☐	policy1234567841	visa	visa_signature	sally_squirrel	2021_01_01	2021_12_31	

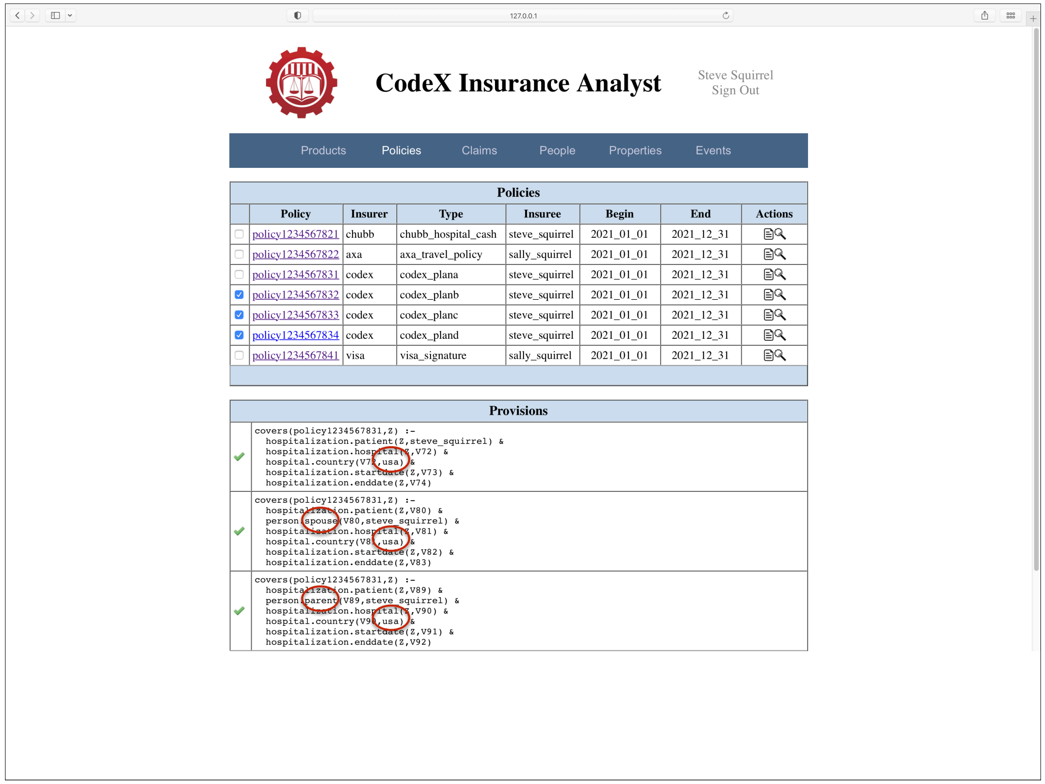
Provisions

✔
```
covers(policy1234567831,Z) :-
  hospitalization.patient(Z,steve_squirrel) &
  hospitalization.hospital(Z,V72) &
  hospital.country(V72,usa) &
  hospitalization.startdate(Z,V73) &
  hospitalization.enddate(Z,V74)
```

✔
```
covers(policy1234567831,Z) :-
  hospitalization.patient(Z,V80) &
  person.spouse(V80,steve_squirrel) &
  hospitalization.hospital(Z,V81) &
  hospital.country(V81,usa) &
  hospitalization.startdate(Z,V82) &
  hospitalization.enddate(Z,V83)
```

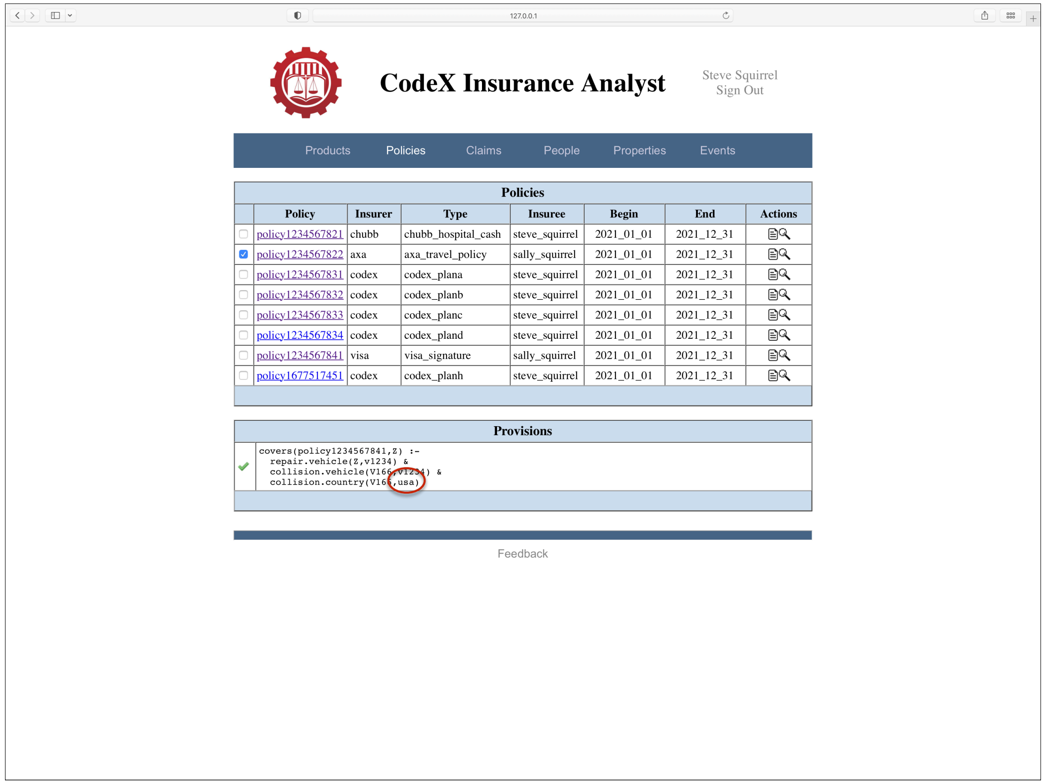
```
covers(policy1234567831,Z) :-
  hospitalization.patient(Z,V89) &
  person.parent(V89,steve_squirrel) &
  hospitalization.hospital(Z,V90) &
  hospital.country(V90,usa) &
  hospitalization.startdate(Z,V91) &
  hospitalization.enddate(Z,V92)
```

Planc covers his spouse anywhere in the world. Checking the planc box leads to a checkmark next to the second provision. On other words, plan b covers the second provision.



Pland covers his kids. Checking the planbd box leads to a checkmark next to the third provision. In other words, plan c covers the first provision. The upshot is that these three policies entirely cover the provisions of plana.

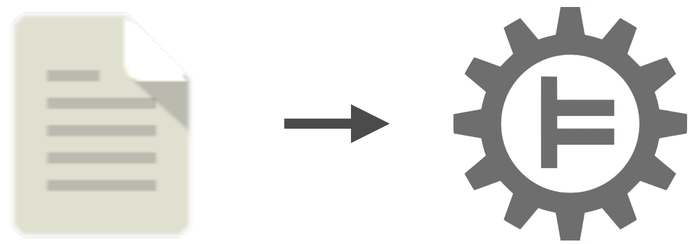
What if we were to reverse the analysis, pressing the magnifier next to plan b and checking the plan a box, nothing would happen in this case, since plan a covers only US hospitlizations, whereas plan b covers hospitalizations anywhere in the world. In the next generation of the system, we expect to offer explanations of gaps like this.



One more quick example. The visa_signature policy provides coverage in the us, while AXA is world-wide. If we ask te system to analyze the visa-signature policy and check the axa-travel_insurance policy, we see that the AXA policy subsumes the visa_signature policy but not vice versa.

There is a lot more analysis one can do here. But hopefully these examples illustrate the basic idea of the Codex Insurance Analyst. Time now to say a few words about the technology.

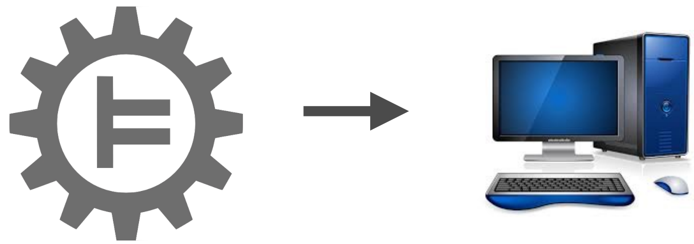
Computable Contracts



From Textual Policies to Computable Contracts

Our approach to preparing creating a system like CIA has two parts. (1) The first part is to formalize the terms and conditions of traditional textual policies as "computable contracts", i.e. contracts specified in sufficient detail that it is possible to unambiguously determine compliance of clearly specified circumstances with the terms and conditions of the contract.

Contract Automation



Automating Insurance Processes

(2) The second step is to use these computable contracts in automating the processes of the insurance ecosystem - thereby bringing the power of computers and networks to bear on provide the capabilities we just saw.

Formal, Declarative Representation

Definitions of Concepts (Ontologies)

A grandparent is a parent of a parent.

```
grandparent(X,Z) :- parent(X,Y) & parent(Y,Z)
```

Constraints

Parents are older than their children.

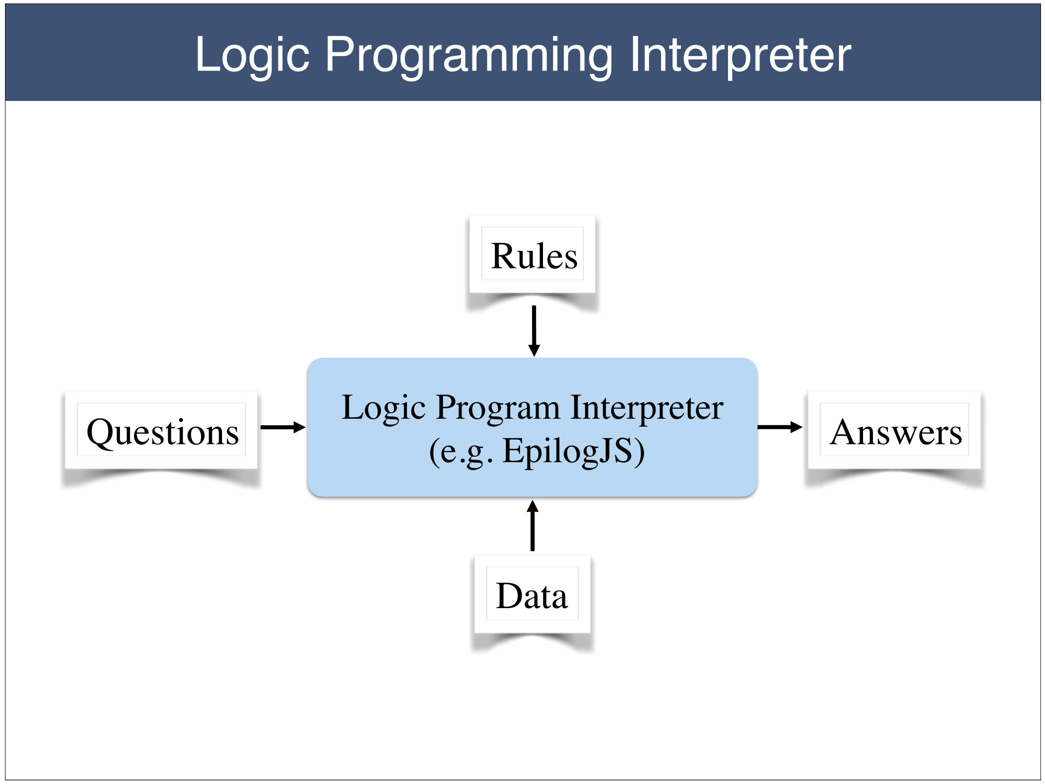
```
illegal :- parent(X,Y) & age(X)<age(Y)
```

Policies - Definitions of Coverage

Policy 1 covers all hospitalizations in the usa.

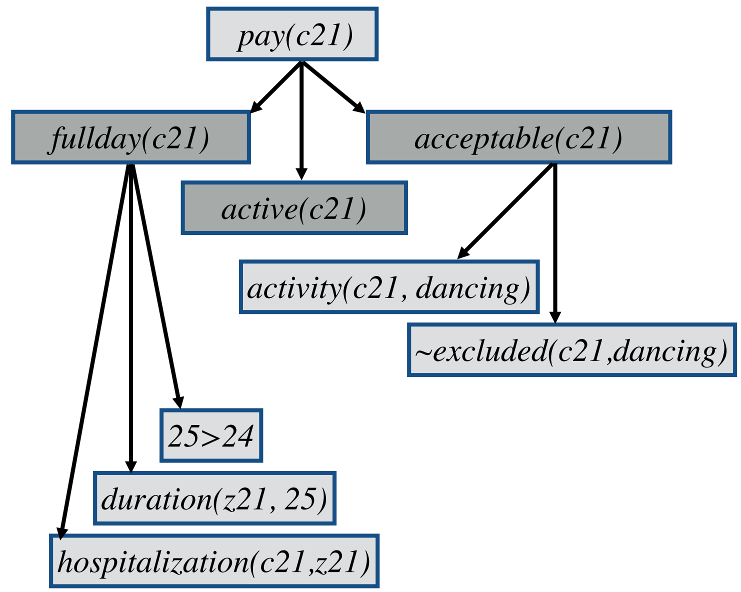
```
covers(policy1,H) :- location(H,usa)
```

Our approach to formalizing and policies and implementing analysis is based on Logic programming. In this approach, we encode information in the language of logic. Using this language, we can define new concepts; we can state physical constraints; and we can encode rules and regulations. A grandparent is a parent of a parent. Parents are older than their children. And a particular policy applies to hospitals in the US. Like natural language, the language of Logic is expressive; yet, unlike natural language, it is grammatically simple and unambiguous.



Moreover, we know how to build interpreters that can reliably answer questions from data and rules, and we can use these interpreters for multiple purposes - whether analyzing individual claims or analyzing whole policies.

Derivation Trees



But wait, there's more. Things do not necessarily end with the derivation of answers. We can also give users explanations. In deriving conclusions, LP systems implicitly or explicitly generate derivation trees for those conclusions, like the one shown here, and it is possible to generate sensible explanations by incrementally exhibiting those derivation trees to the user. Claim 21 should be paid because it meets the full day requirement and the policy is active and the activity that caused the hospitalization is not excluded. The claim meets the full day requirement because the underlying hospitalization had a duration of greater than 24 hours. The claim is acceptable because the hospitalization was due to a dancing mishap and that is not excluded.

Template-Based NL Generation

Templates:

`fullday(X)`	→	*"Claim X is full day"*
`hospitalstay(X,Z)`	→	*"the hospital stay of X is Z"*
`duration(Z,H)`	→	*"the duration of Z is H hours"*
`H>J`	→	*"H is greater than J"*

Derivation Tree:

```
fullday(c21) :-
 hospitalstay(c21,z21)&
 duration(z21,25)&
 25>24
```

Explanation:

Claim c21 is full day because
the hospital stay of c21 is z21 and
the duration of z21 is 25 hours and
25 is greater than 24.

In progress: Pronouns, prepositional phrases, relative clauses

Such explanations can be rendered into sensible (if not ideal) form using templates to generate snippets of natural language like the ones shown here. There are subtleties in explanation and conversion to natural language, but this approach renders comprehensible and accurate explanations relatively easily.

Contract Definition Language (CDL)

View Definition Language (Epilog)
terms of coverage in terms of domain concepts
definitions of concepts in terms of available data

Contract / Insurance Ontology (in progress)
offer, acceptance, consideration, etc.
premiums, claims, claim lines, deductibles, exclusions

Domain Ontology and Data (in progress)
e.g. diseases, drugs, medical procedures, etc.
e.g. fenders, radiators, repair procedures, etc.
e.g. walls, external water, internal water, etc.

Key to our approach is a contract definition language of use in codifying computable contracts. There are three parts to the language - (1) a *view definition* language, (2) a *contract ontology*, and (3) a *domain ontology*. (The view definition language is loosely analogous to the grammar of English; the contract ontology consists of words for legal concepts; and the domain ontology consists of words for the subject matter or contracts.)

The view definition language provides a way for users to define higher level concepts in terms of lower level concepts. In this regard, we propose to use an off-the-shelf Logic Programming Language. Over the years, a number of different Logic Programming languages have been developed, e.g. Datalog, Prolog, and Epilog. In our work, we propose to use Epilog, as it provides greater expressiveness than Datalog and has a cleaner semantics than Prolog. Also, we have used it effectively in the past in various Computational Law applications and in exploratory work on formalizing contracts.

The contract ontology is a collection of concepts related to contracts in general - notions such as meeting of the minds, offer and acceptance, consideration, and so forth. We propose to establish vocabulary for naming such concepts so that those components of contracts can be recorded explicitly. While that this ontology is essential for ensuring that contracts are valid, it is unnecessary for determining compliance of situations with the terms and conditions of contracts. For that we need a domain ontology.

The domain ontology is a collection of concepts relevant to subject matter of contracts. For example, in the case of health insurance, it would contain words for diseases and drugs and medical procedures. In the case of car insurance, it would contain words for car parts and repair techniques. For homeowners insurance, it would contain words for foundations and walls and finishes.

Of these three, the domain ontology is requires the most work. Luckily, there are emerging standards in a variety of areas. For example, in the medical arena, there are formal classifications of diseases and drugs and procedures; and there are data bases knowledge bases defining these concepts in terms of each other, e.g. units and currency conversions. Our work will be in (1) finding suitable sources of ontologies, databases, and knowledge bases, (2) assessing their suitability for encoding related contracts to computable form, and (3) relating these ontologies to the data schemas used by organizational database systems containing information that must be accessed to determine compliance with the terms and conditions of contracts.

Contract IDE

In addition to CDL and appropriate interpreters, we are using a variety of tools to facilitate the authoring, revision, and debugging of computable policies. The computer programming community has long recognized the value of Interactive Development Environments (IDEs) to help in developing and maintaining programs. The use of LP makes possible the development of especially powerful IDEs for authoring, analyzing, and debugging policies as well.

Contractnet

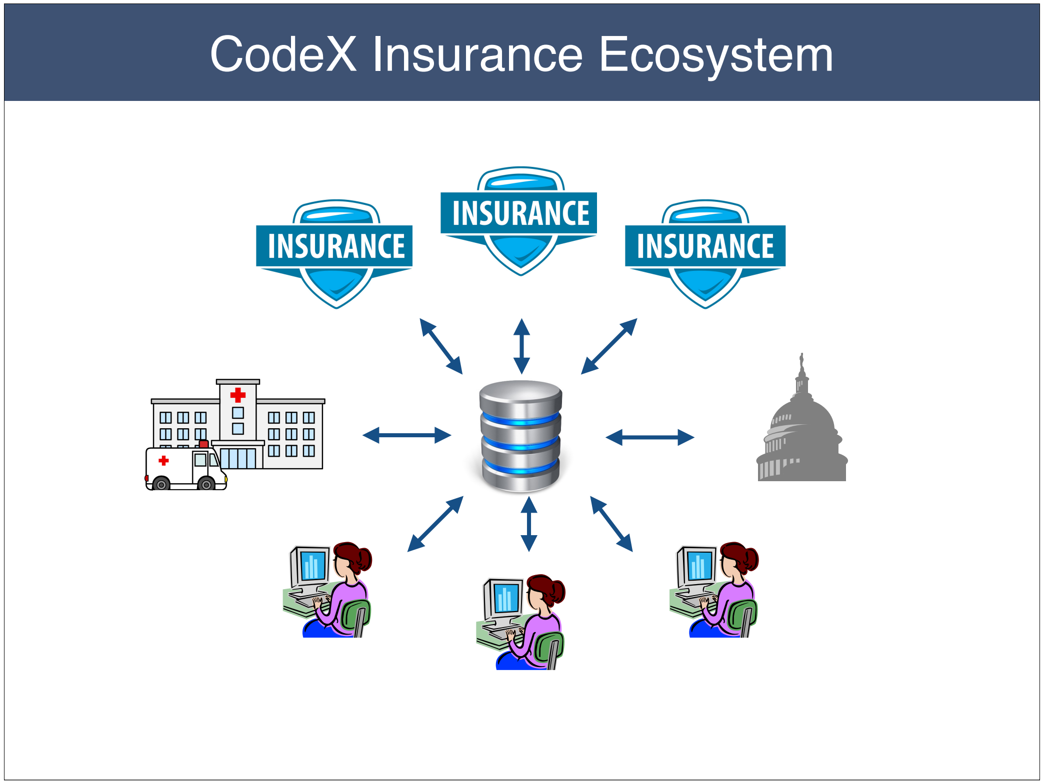
Background Datasets
Geography
Hospitals

Background Rulesets
e.g. a week is 7 days
e.g. parents are older than their children

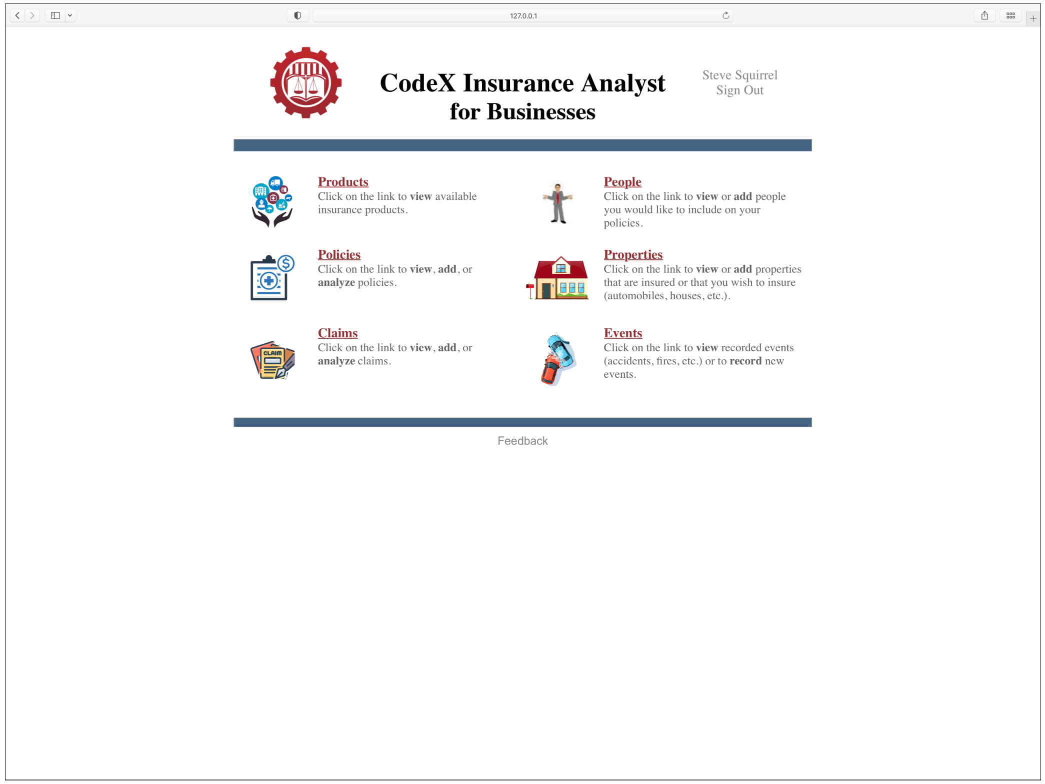
Open-Source Contract Processing Software
Compliance checking (EpilogJS)
Explaining contractual analysis (ExplainJS, EnglishJS)
User Worksheets (WorksheetsJS)
Containment Testing and Configuration Aids (TBD)

Github for Computable Insurance Contract Software

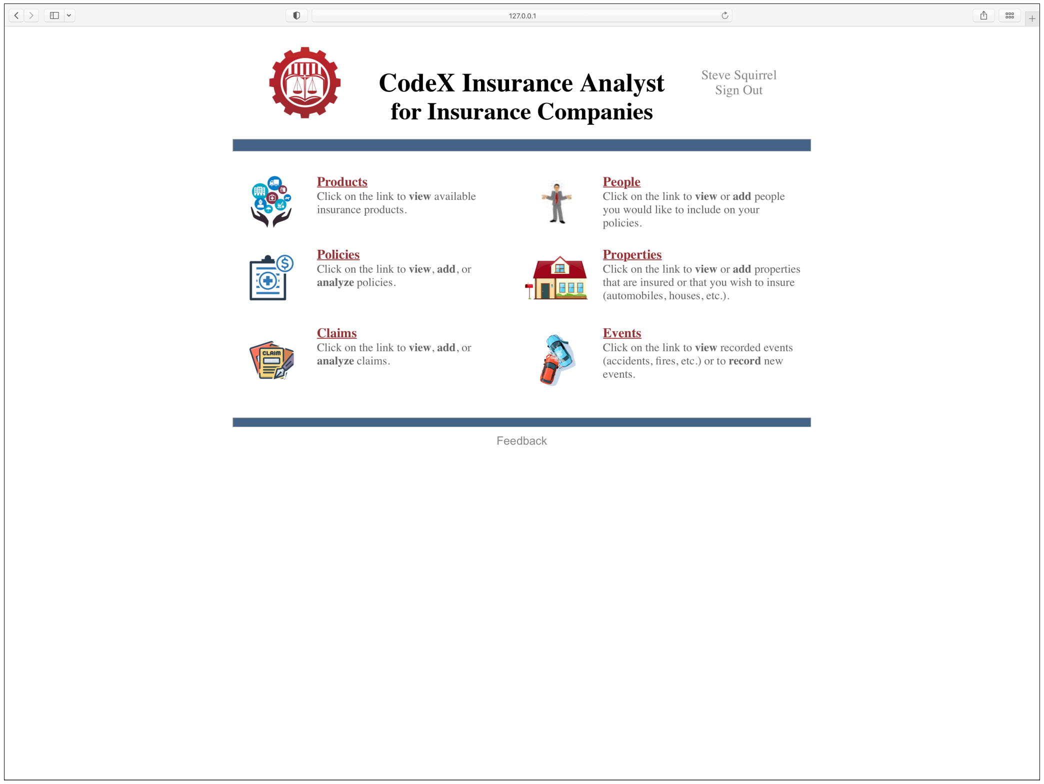
In order to promote the use of this technology, we propose to develop a repository of datasets, rulesets, and software for everyone to use, in a system called Contractnet. GITHUB is a popular Internet site for programmers to use in sharing open-source code with others. We think of Contractnet as the GITHUB for computable contracts.



Ultimately, we believe that automated Insurance Portfolio Management can provide benefit for all parties in the insurance ecosystem. It can benefit customers by analyzing their overall coverage and synthesizing portfolios of insurance products that meets their needs with minimal cost. It can help insurance companies increase transparency, decrease meritless claims, detect upsell opportunities, and craft customized insurance products. And it can help regulators support and enforce insurance regulations.



Although the current prototype is focussed on individuals, we believe that a similar system can be built to aid businesses in analyzing their insurance needs and opportunities.



And we think that insurance companies can benefit from analyzing their portfolios to understand their risks and business opportunities.

CODEX

The Stanford Center for Legal Informatics

Legal Empowerment through Information Technology

The idea of automated insurance portfolio management is an appealing one, and our work thus far suggests that it is possible to build useful systems of this sort. There is still more work to be done. However, we would like to move some of this work from the laboratory to the real world. We need to see whether it works in practice; and, if not, we need to understand its weaknesses. Luckily, things are moving forward here. Insurance companies, such as AXA, understand the potential benefits of computable insurance contracts, and they are actively investigating way in which they can utilize this technology in their businesses.

Thank you. Questions?