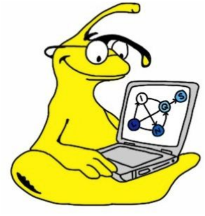
An Introduction to Probabilistic Soft Logic

Eriq Augustine and Golnoosh Farnadi
UC Santa Cruz
MLTrain 2018



psl.linqs.org
github.com/linqs/psl

Probabilistic Soft Logic (PSL) Overview

- Declarative probabilistic programming language for structured prediction
  - Scalable -- inference in PSL is highly efficient
  - Interpretable -- models are specified as weighted rules
  - Expressive -- can model complex dependencies, latent variables, handle missing data
- Open-source: psl.linqs.org

PSL Key Capabilities

- Rich representation language based on logic allows
  - Declarative representation of models
  - Well-suited to domains with structure (e.g., graphs and networks)
- Probabilistic Interpretation
  - Supports uncertainty and “soft” logic
  - Semantics defined via specific from of graphical model referred to as a *Hinge-loss Markov Random Field*

PSL Application Types

- Effective on wide range of problem types
  - data integration, information fusion, & entity resolution
  - recommender systems & user modeling
  - computational social science
  - knowledge graph construction

PSL Sample Application Domains

- Competitive Diffusion in Social Networks
  - Broecheler et al., SocialCom10
- Social Group Modeling
  - Huang et al., Social Networks and Social Media Analysis Workshop NIPS12
- Demographic Prediction & Knowledge Fusion for User Modeling
  - Farnadi et al., MLJ17
- Inferring Organization Attitudes in Social Media
  - Kumar et al., ASONAM16
- Modeling Student Engagement in MOOCs
  - Ramesh et al., AAAI13; Ramesh et al., L@S14; Tomkins et al. EDM16
- Personalization and Explanation in Hybrid Recommender Systems
  - Kouki et al., RecSys15; Kouki et al., RecSys17
- Detecting Cyberbullying in Social Media
  - Tomkins et al., ASONAM

Outline

- Basic Introduction to PSL
- Getting Started with PSL
- PSL Examples
  - Collective Classification
  - Link Prediction
  - Entity Resolution
  - Knowledge Graph Construction
- Conclusion

Why Collective Classification?

Weather Forecasting

Goal: Predict the probability of rain in Santa Cruz.

VS

Local Signals for Prediction

Local sensors provide useful signals for prediction.

Relational Signals for Prediction

Sensors in nearby cities provide useful relational information.

San Jose

32 Miles

Santa Cruz

?

Relational Signals for Prediction

Sensors in nearby cities provide useful relational information.

Weather Forecasting

What if we wanted to predict for multiple cities?

San Jose

32 Miles

Santa Cruz

Diagram for Weather Forecasting

Diagram for Weather Forecasting

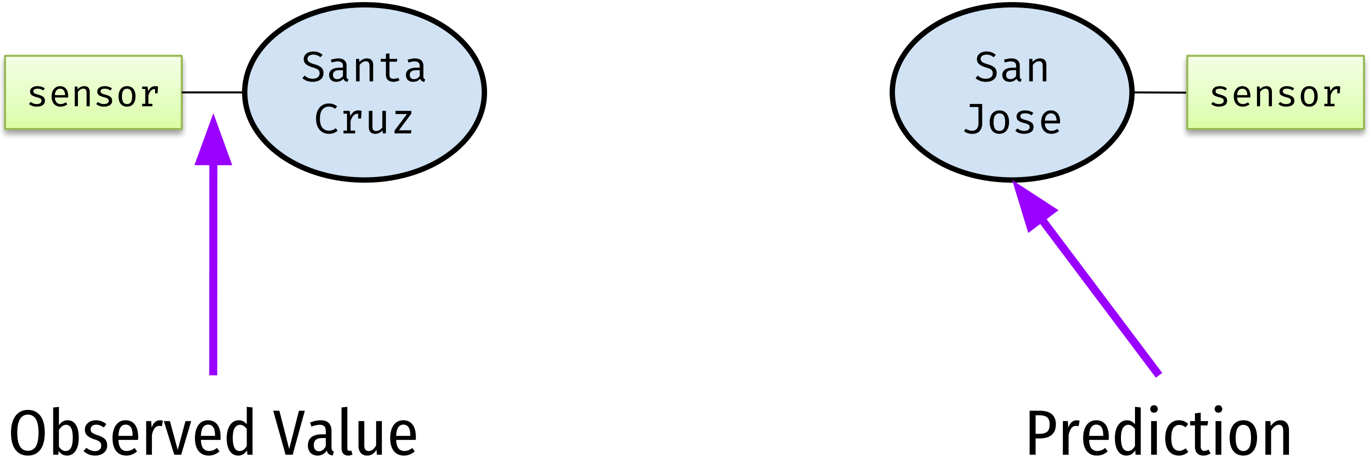
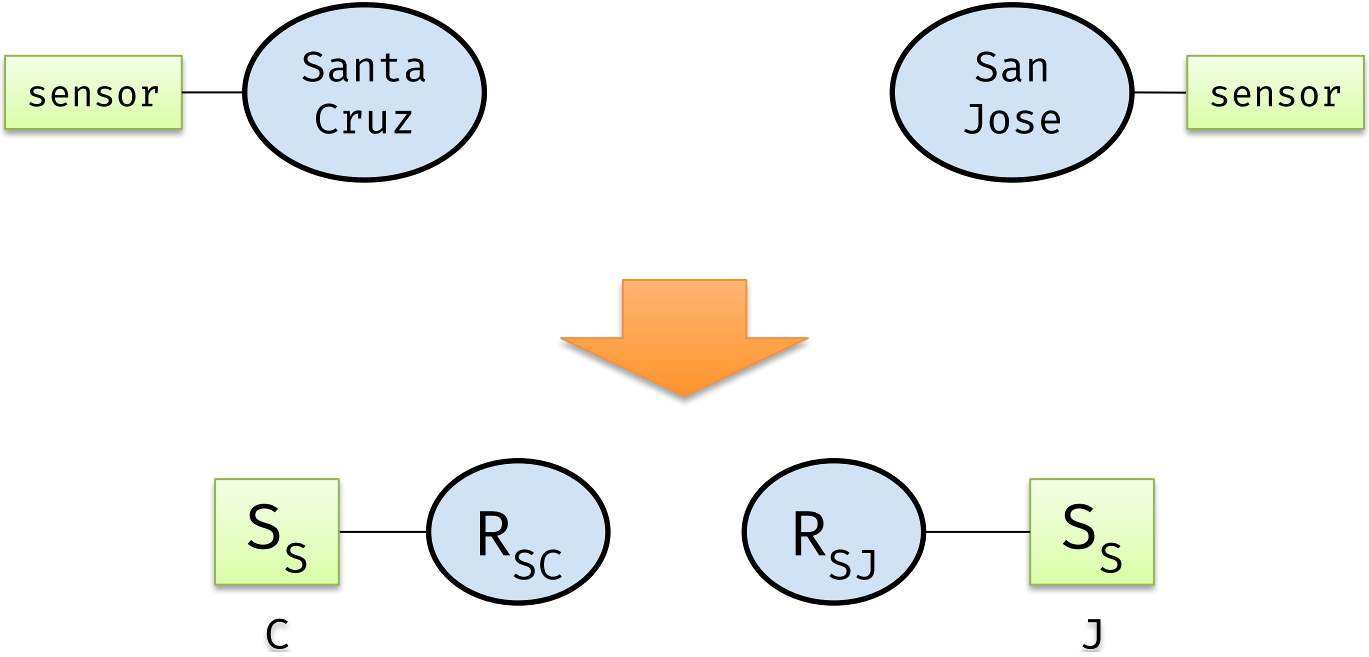
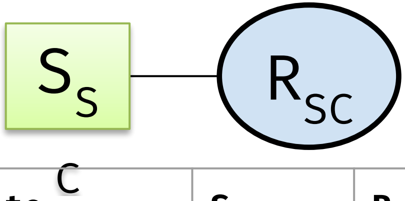


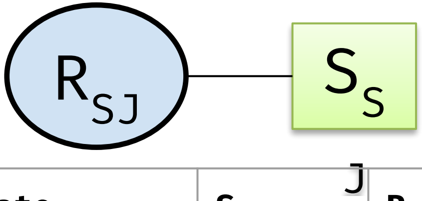
Diagram for Weather Forecasting



Local Predictive Model

Using historical data, we learn independent models for each city.





Date	S_{SC}	R_{SC}
1950-06-06	22.2°C	0
1951-06-06	17.1°C	1
...	...	...
2017-06-06	23.4°C	0

Date	S_{SJ}	R_{SJ}
1950-06-06	25.0°C	0
1951-06-06	20.1°C	1
...	...	...
2017-06-06	24.5°C	0

$$\Pr(R_{SC} | S_{SC})$$

$$\Pr(R_{SJ} | S_{SJ})$$

Incorrect Sensor Reading

Common problem: we get a faulty sensor reading.

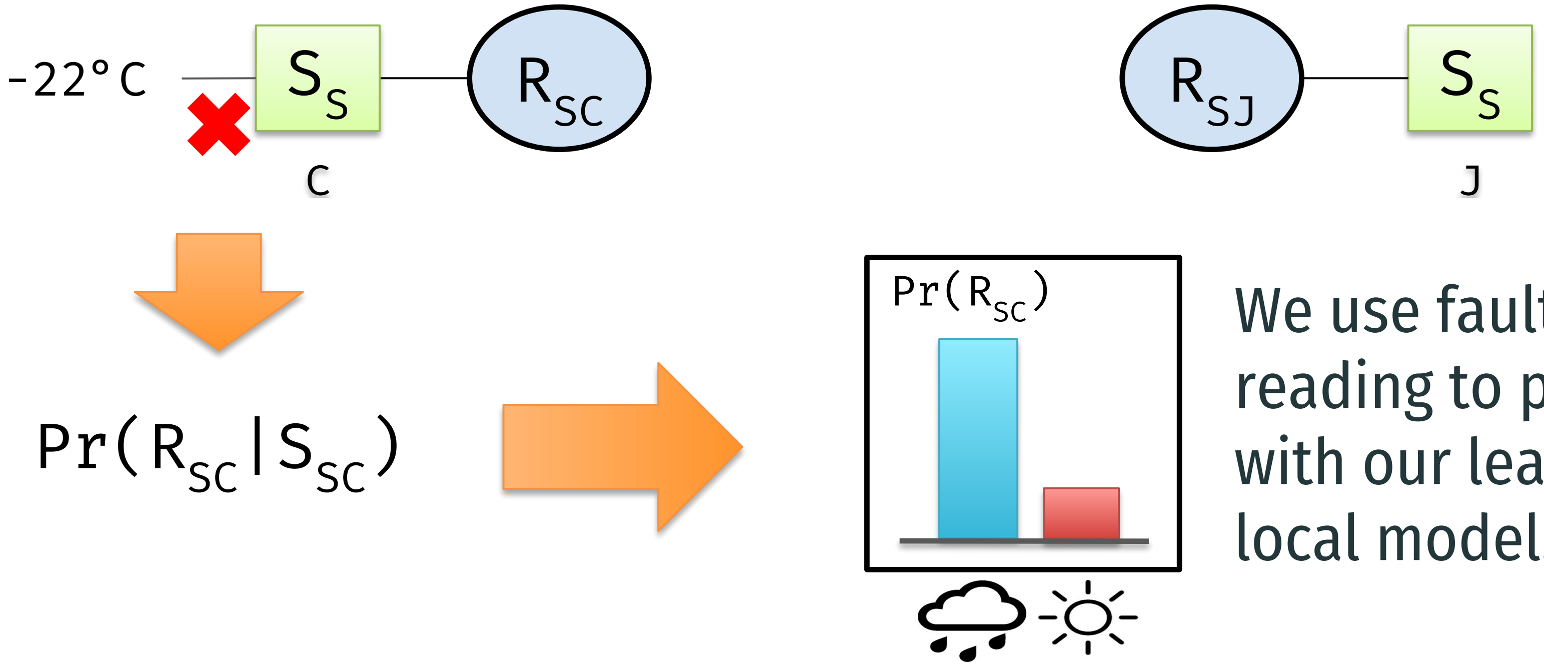
-22° C — S_S

c

Santa Cruz

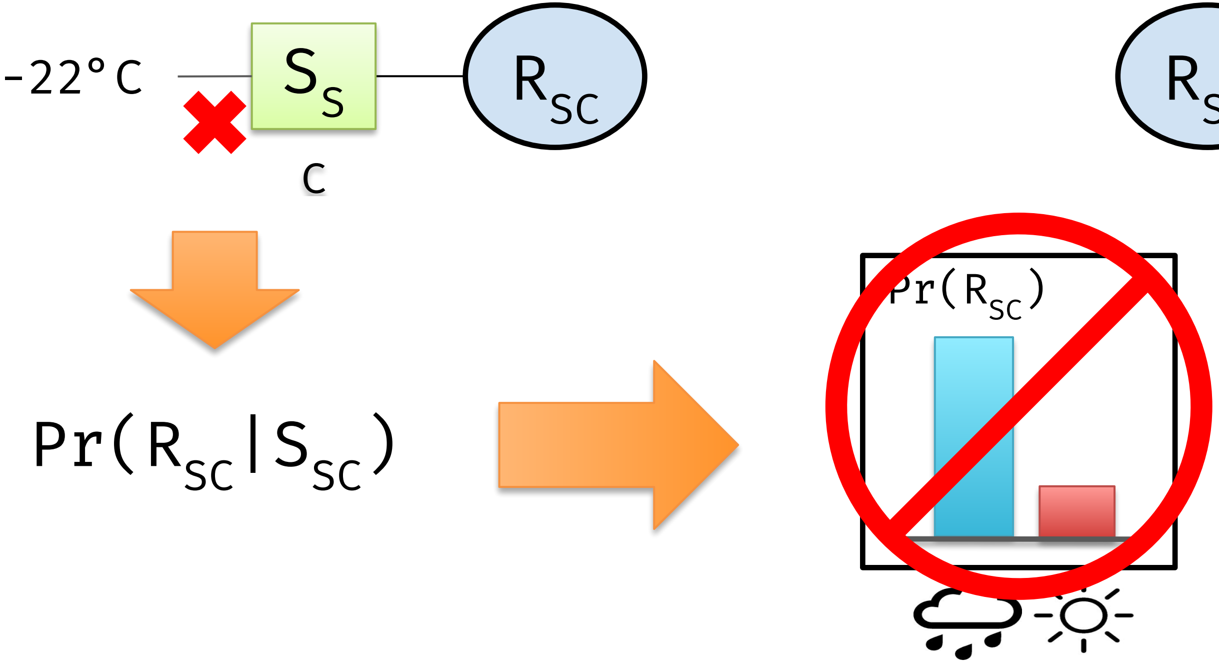
Incorrect Local Predictions

Incorrect Local Predictions



We use faulty reading to predict with our learned local model.

Incorrect Local Predictions



Common outcome: local model makes incorrect prediction.

Relational Signals for Prediction

Recall: sensors in nearby cities provide useful relational information!

San Jose

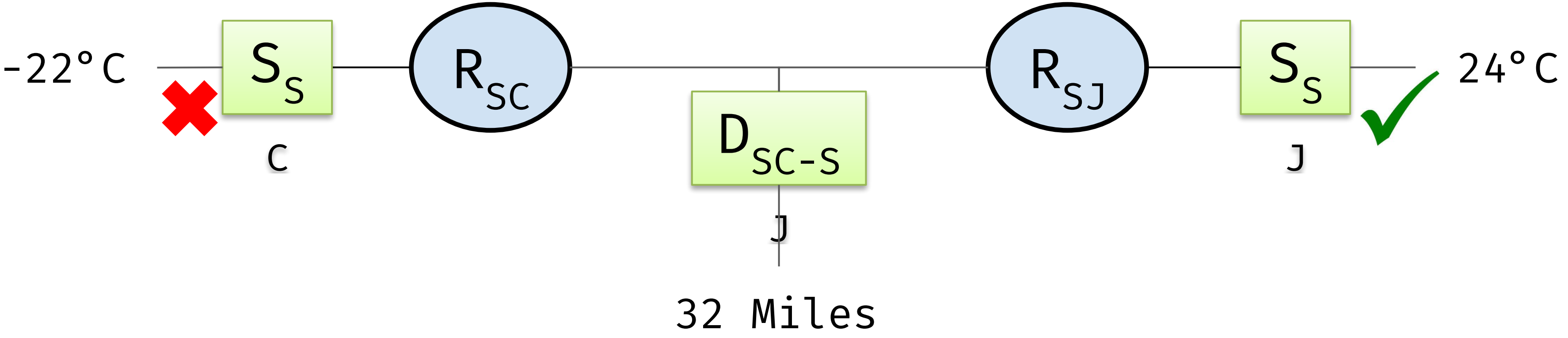
32 Miles

Santa Cruz

Leveraging Relational Signals

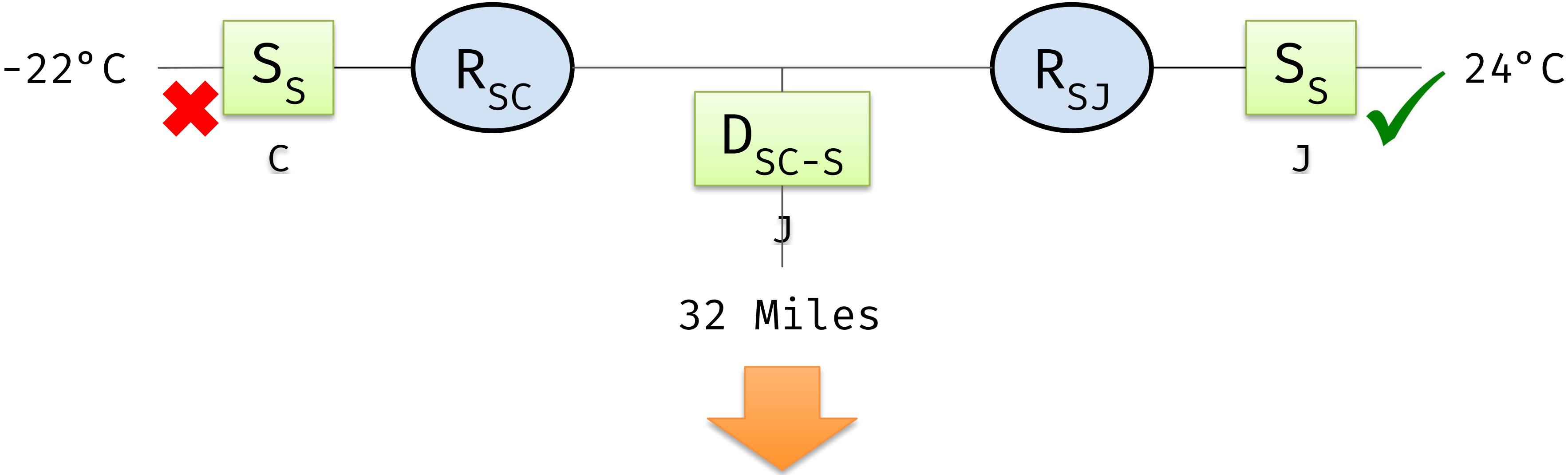
Leveraging Relational Signals

Distance variable captures closeness between cities.



Leveraging Relational Signals

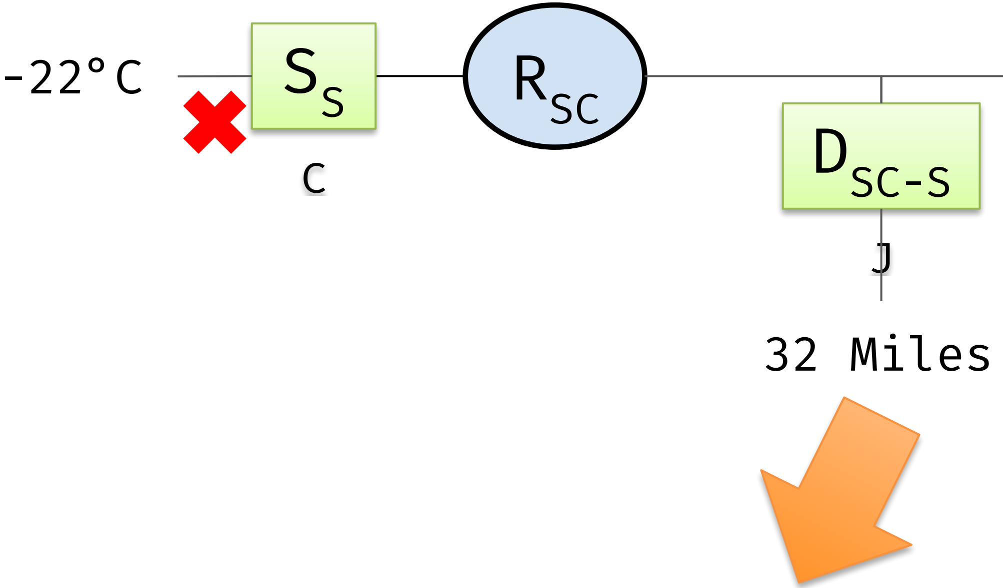
Distance variable captures closeness between cities.

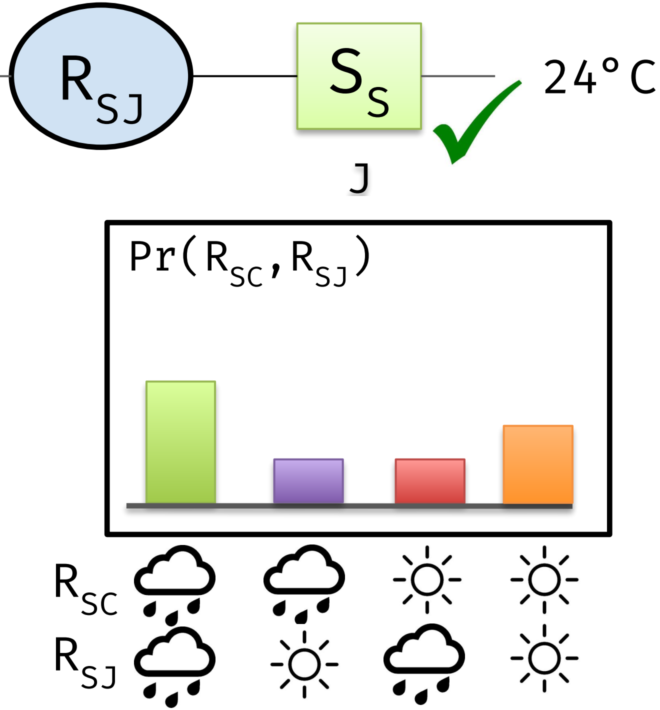


$$Pr(R_{SC}, R_{SJ} | S_{SC}, S_{SJ}, D_{SC-SJ})$$

Leveraging Relational Signals

Joint modeling: forecasts in nearby cities should be similar.

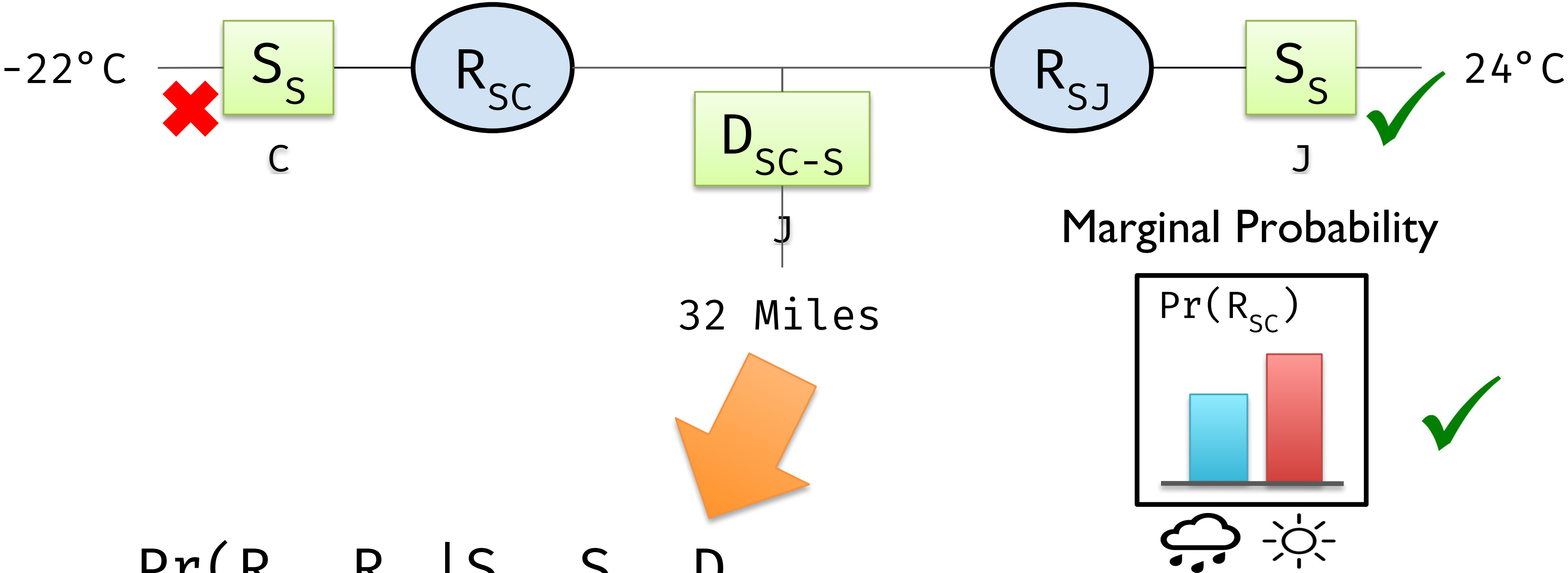




$$Pr(R_{SC}, R_{SJ} | S_{SC}, S_{SJ}, D_{SC-SJ})$$

Leveraging Relational Signals

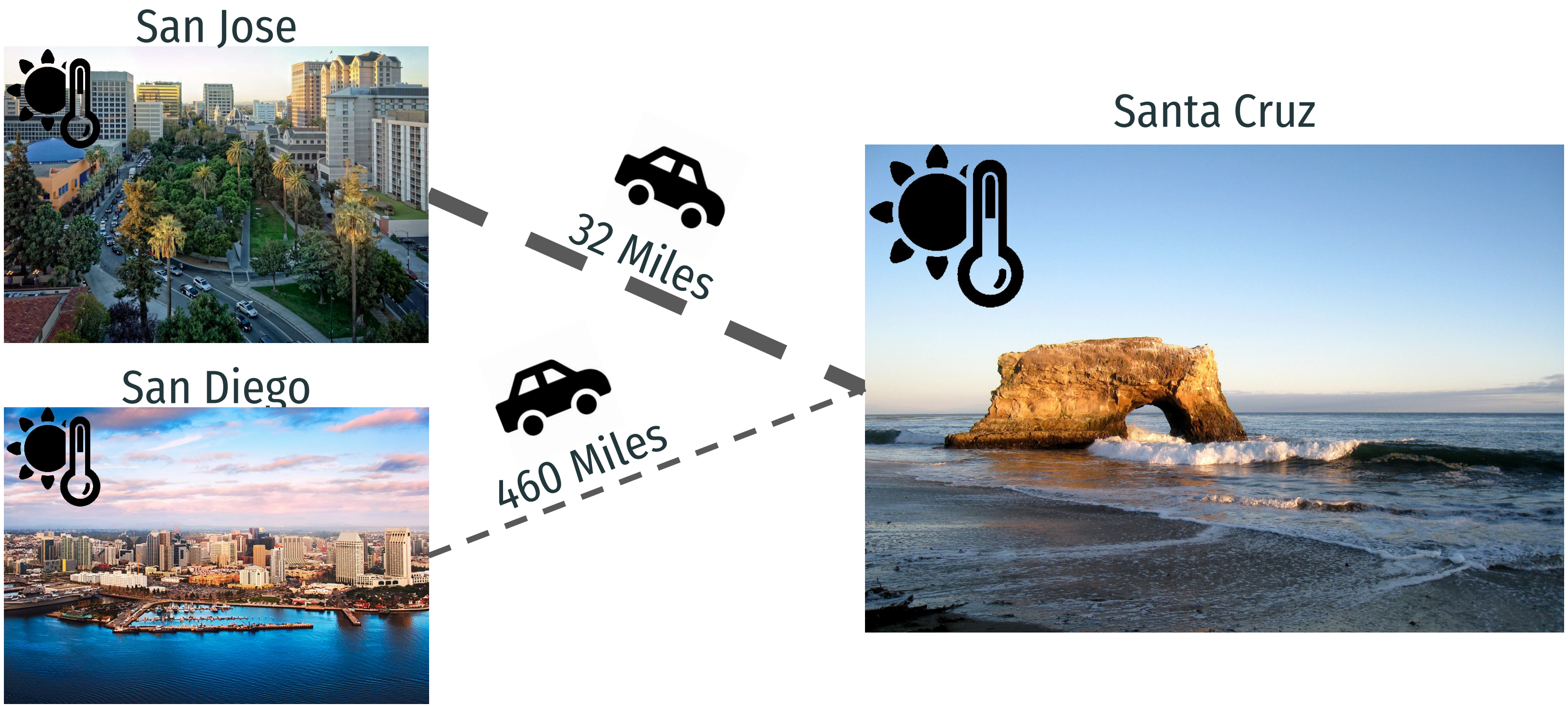
Joint modeling: forecasts in nearby cities should be similar.



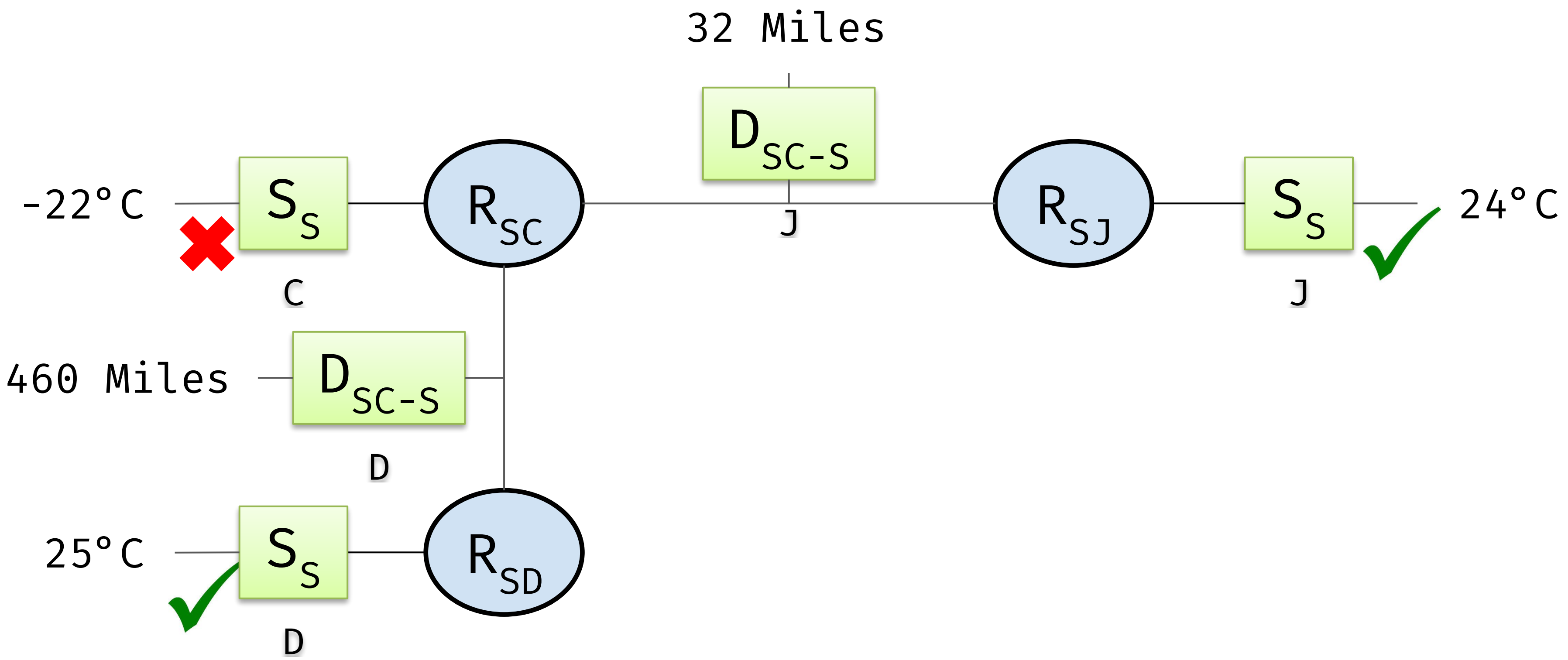
$$Pr(R_{SC}, R_{SJ} | S_{SC}, S_{SJ}, D_{SC-SJ})$$

Combining Multiple Relational Signals

Nearby cities should have a greater relational influence than far away cities.

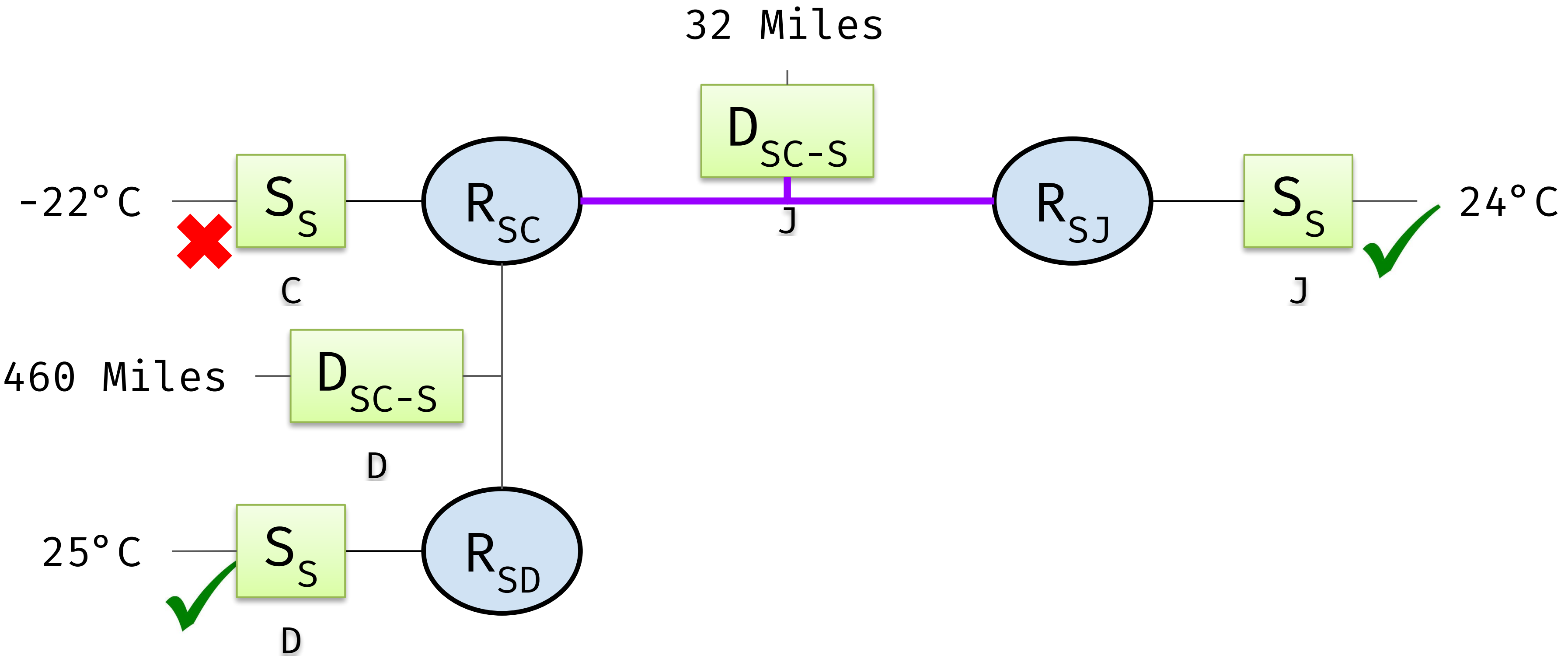


Relative Influences of Neighbors



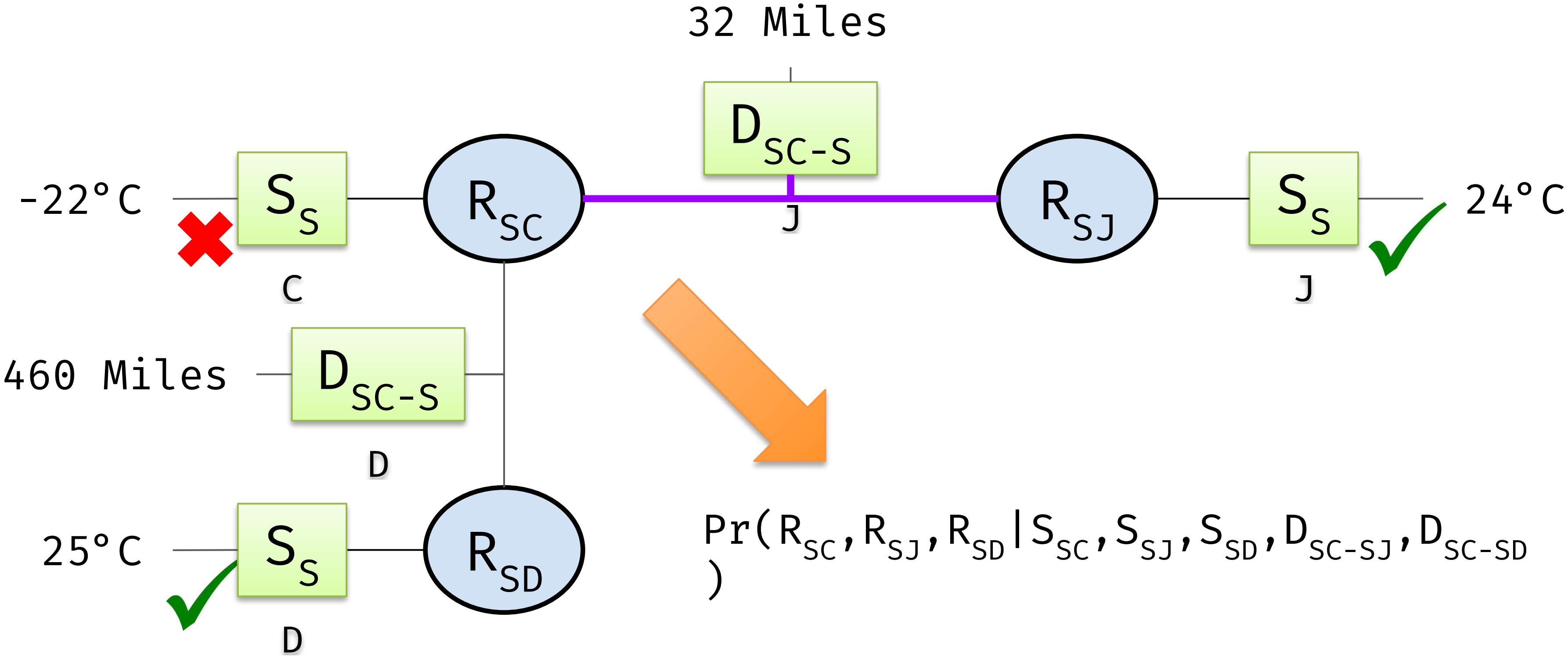
Relative Influences of Neighbors

Strength of collective influence depends on distance between cities.



Relative Influences of Neighbors

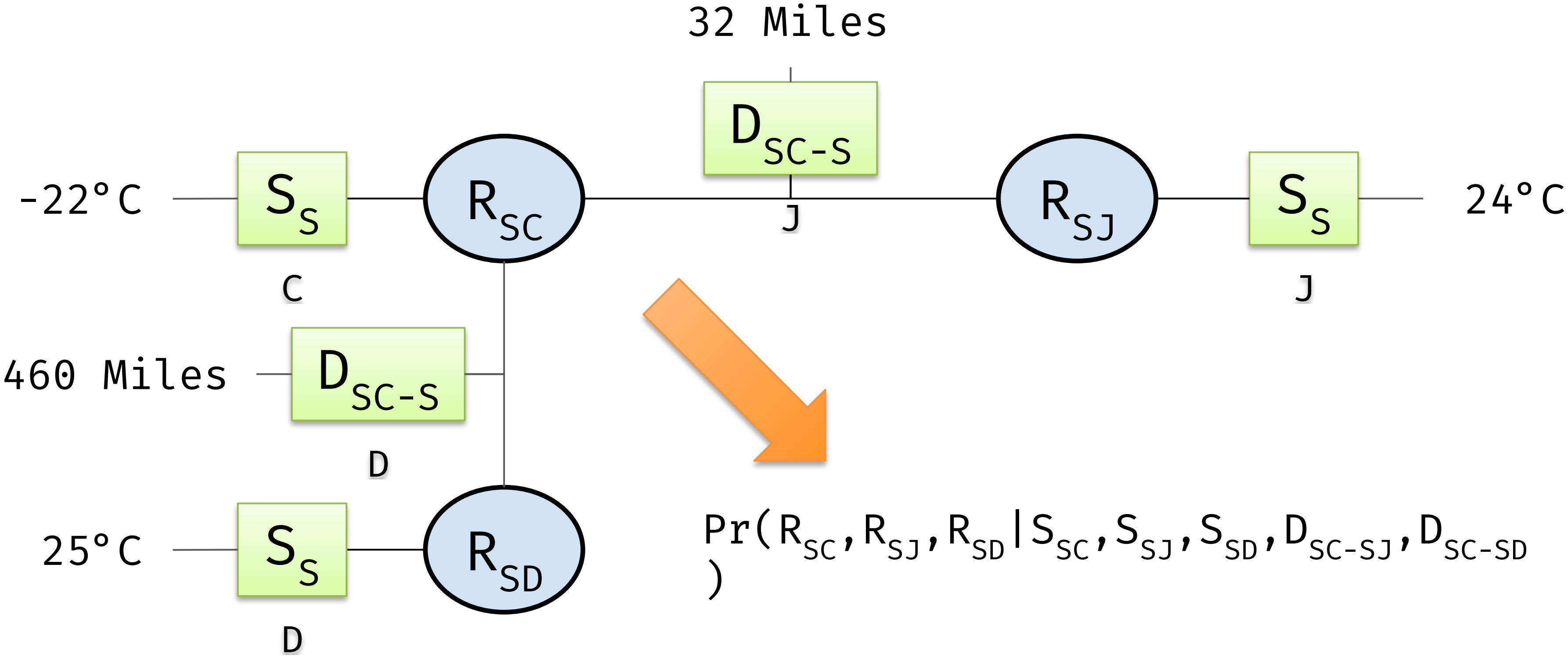
Distance variables $D_{SC\text{-}SJ}$ and $D_{SC\text{-}SD}$ mediate affinity of forecasts between cities.



$$\Pr(R_{SC}, R_{SJ}, R_{SD} \mid S_{SC}, S_{SJ}, S_{SD}, D_{SC\text{-}SJ}, D_{SC\text{-}SD})$$

Markov Random Fields (MRFs)

This graphical model is a Markov Random Field (MRF).



$Pr(R_{SC}, R_{SJ}, R_{SD} | S_{SC}, S_{SJ}, S_{SD}, D_{SC-SJ}, D_{SC-SD})$

PSL - Syntax and Semantics

PSL

PSL uses first order logic-like rules.

```
5.0: Rainy(City1) & Distance(City1, City2) -> Rainy(City2)
1.0: SenseRain(City)                       -> Rainy(City)
```

PSL

PSL uses first order logic-like rules.

```
5.0: Rainy(City1) & Distance(City1, City2) -> Rainy(City2)
1.0: SenseRain(City)                       -> Rainy(City)
```

Weight

Predicate

Variable

PSL - Templating Language for MRFs

```
5.0: Rainy(City1) & Distance(City1, City2) -> Rainy(City2)
```

```
1.0: SenseRain(City) -> Rainy(City)
```

PSL - Templating Language for MRFs

Rule templates instantiated with data become "Ground Rules".

```
5.0: Rainy(City1) & Distance(City1, City2) -> Rainy(City2)
```

```
5.0: Rainy('Cruz') & Distance('Cruz', 'Jose') -> Rainy('Jose')
5.0: Rainy('Cruz') & Distance('Cruz', 'Diego') -> Rainy('Diego')
```

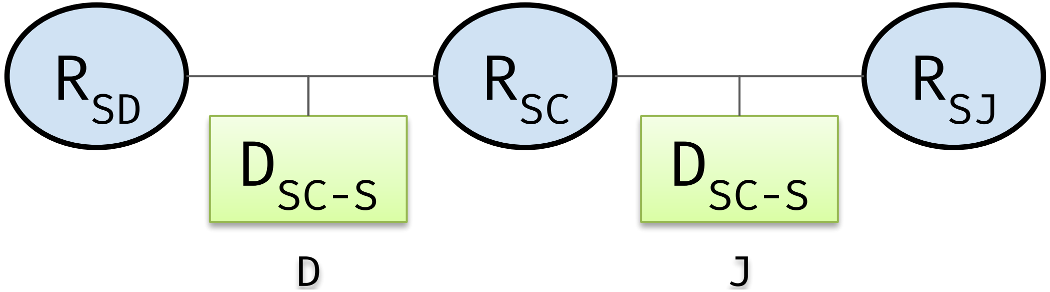
```
1.0: SenseRain(City) -> Rainy(City)
```

```
1.0: SenseRain('Cruz') -> Rainy('Cruz')
1.0: SenseRain('Jose') -> Rainy('Jose')
1.0: SenseRain('Diego') -> Rainy('Diego')
```

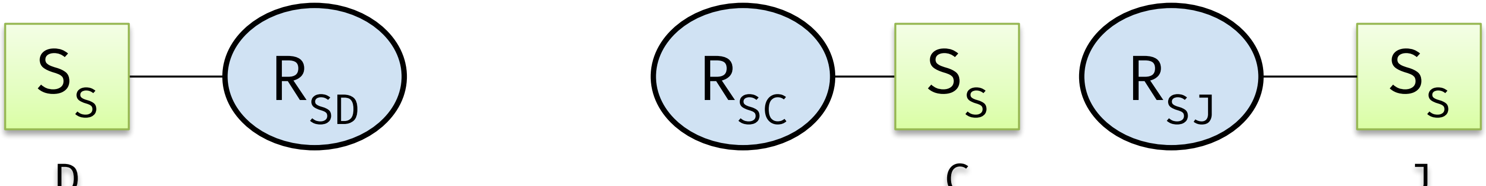
PSL - Templating Language for MRFs

Ground rules directly map to potential functions in the MRF.

```
5.0: Rainy(City1) & Distance(City1, City2) -> Rainy(City2)
```

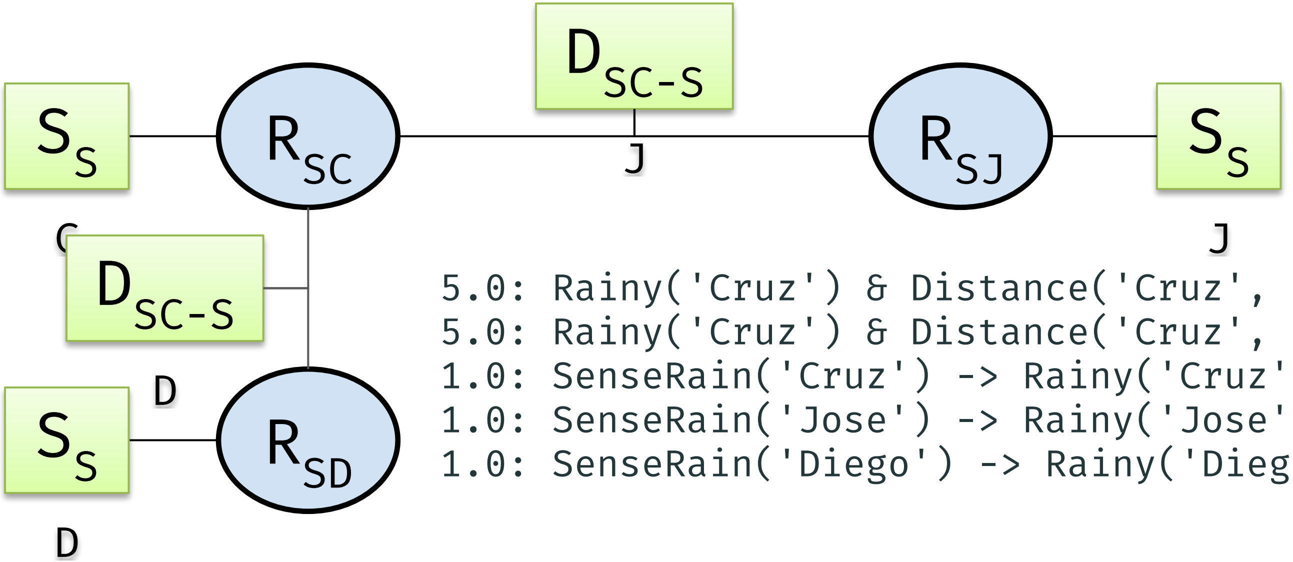



```
1.0: SenseRain(City)                             -> Rainy(City)
```



PSL - Templating Language for MRFs

```
5.0: Rainy(City1) & Distance(City1, City2) -> Rainy(City2)
1.0: SenseRain(City)                       -> Rainy(City)
```




```
5.0: Rainy('Cruz') & Distance('Cruz', 'Jose') -> Rainy('Jose')
5.0: Rainy('Cruz') & Distance('Cruz', 'Diego') -> Rainy('Diego')
1.0: SenseRain('Cruz') -> Rainy('Cruz')
1.0: SenseRain('Jose') -> Rainy('Jose')
1.0: SenseRain('Diego') -> Rainy('Diego')
```

PSL - MRF Inference

$$P(Y|X) \propto exp(\sum_{i}^{G} w_i \phi_i)$$

Sum over all ground rules.

The weight for a rule.

The "satisfaction" of a ground rule. 1/0 for discrete logic.

PSL - MRF Inference

$$P(Y|X) \propto exp(\sum_{i}^{G} w_i \phi_i)$$

$$\mathrm{argmax}_X \sum_{i}^{G} w_i \phi_i$$

PSL - MRF Inference

Discrete MRF Inference == Weighted MAX-SAT == NP-Hard

$$\text{argmax}_X \sum_i^G w_i \phi_i$$

PSL - Continuous Relaxation

Relax "hard" satisfiability of each rule.

```
5.0: Rainy(City1) & Distance(City1, City2) -> Rainy(City2)
```

PSL - Continuous Relaxation

First convert the rule to Disjunctive Normal Form.

```
5.0: Rainy(City1) & Distance(City1, City2) -> Rainy(City2)
```

Rainy(City1) ^ Distance(City1, City2) -> Rainy(City2)

¬(Rainy(City1) ^ Distance(City1, City2)) v Rainy(City2)

¬Rainy(City1) v ¬Distance(City1, City2) v Rainy(City2)

PSL - Continuous Relaxation

Use Łukasiewicz logic to relax hard logical operators.

- `P ^ Q = max(0.0, P + Q - 1.0)`
- `P v Q = min(1.0, P + Q)`
- `¬Q = 1.0 - Q`

PSL - Continuous Relaxation

Apply Łukasiewicz logic.

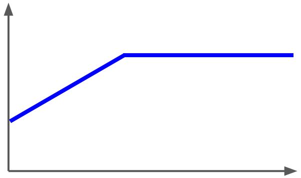
```
¬Rainy(City1) v ¬Distance(City1, City2) v Rainy(City2)

min(1.0, ¬Rainy(City1) + ¬Distance(City1, City2)) v Rainy(City2)

min(1.0, ¬Rainy(City1) + ¬Distance(City1, City2) + Rainy(City2)

min(1.0, (1.0 - Rainy(City1)) + (1.0 - Distance(City1, City2))
    + Rainy(City2))
```

`min(1.0, 2.0 - (Rainy(City1) + Distance(City1, City2))`
`    + Rainy(City2))`

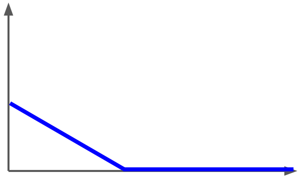
PSL - Continuous Relaxation

Apply Łukasiewicz logic to form a Hinge-Loss MRF.

Satisfaction:
`min(1.0, 2.0 - (Rainy(City1) + Distance(City1, City2)) + Rainy(City2))`



Distance to satisfaction:
`1.0 - min(1.0, 2.0 - (Rainy(City1) + Distance(City1, City2)) + Rainy(City2))`



PSL - HL-MRF Inference

HL-MRF Inference == Sum of Convex Function == Convex!

Solve with Alternating Direction Method of Multipliers (ADMM)

$$\text{argmax}_X \sum_{i}^{G} w_i \phi_i$$

https://web.stanford.edu/~boyd/admm.html

PSL - Rules to Assignments