modelling HTTP

Daniel Jackson
basic object model for HTTP
in Alloy

```alloy
abstract sig EndPoint {
  causes: set HTTPEvent
}
sig Client, Server extends EndPoint {}
abstract sig HTTPEvent {
  from, to: EndPoint
}
sig Request extends HTTPEvent {
  response: Response
}
sig Response extends HTTPEvent {
  embeds: set Embedded
}
sig Embedded extends Request {}
sig Redirect extends Response {}
run {some Request and some Response}
```
abstract sig EndPoint {
  causes: set HTTPEvent
}
sig Client, Server extends EndPoint {}
abstract sig HTTPEvent {
  from, to: EndPoint
}
sig Request extends HTTPEvent {
  response: Response
}
sig Response extends HTTPEvent {
  embeds: set Embedded
}
sig Embedded extends Request {}
sig Redirect extends Response {}
run {some Request and some Response}
adding constraints

```plaintext
abstract sig EndPoint { causes: set HTTPEvent }

{ causes = {e: HTTPEvent - Embedded | e.from = this} + causes.embeds }

sig Client, Server extends EndPoint {}

abstract sig HTTPEvent {from, to: EndPoint}

sig Request extends HTTPEvent { response: Response }

{ from in Client and to in Server }

sig Response extends HTTPEvent { embeds: set Embedded }

{ from in Server and to in Client }

sig Embedded extends Request {}

fact {Embedded = Response.embeds}

sig Redirect extends Response {}

fact RequestResponse {
    response in Request one -> one Response
    all r: Request | r.from = r.response.to and r.to = r.response.from
}

run {some Request and some Response}
```
another instance

Redirect from: Server
to: Client

causes

Embedded from: Client
to: Server

causes

response
embeds

Client

oops!
adding timing

abstract sig HTTPEvent {
  from, to: EndPoint
  at: Time,
  exists: set Time
}

fact Timing {
  all e: HTTPEvent {
    e.exists = e.at + e.at.nexts
    e.(embeds+response).at in e.at.nexts
  }
}