U0CL0AS3V : so you could presumably do something like
U0CL0AS3V : `ProjectStructure user company contact -> String`
U2GTQM83A : Yes!
U0CL0AS3V : and inside it pattern matches on the argument 3 times
U2GTQM83A : That's what I want.
U0CL0AS3V : first one looks like this
U0CL0AS3V : oh, sorry
U0CL0AS3V : `type alias`
U0CL0AS3V : hmm
U2GTQM83A : And here is the thing. If I'm decoding `Project String String String` I don't need to embed anything. But if I'm decoding `Project User String String` I need to embed "created_by" and "owner".
U0CL0AS3V : so if you refactor to this:
```
type ProjectStructure user company contact =
  ProjectStructure user company contact
    { id : String
    , code : String
    , name : String
    , description : String
    , type_ : ProjectType
    , startDate : Date
    , endDate : Maybe Date
    , budget : String
    , requiredFiles : List String
    , notes : String
    , company : company
    , owner : user
    , contacts : List (ProjectContact contact)
    , rates : List PositionRate
    , created_by : user
    , confidential : Bool
    }
```

U0CL0AS3V : then you can pattern match on it
U2GTQM83A : But pattern match like what? because the user itself may be a parameterised type.
U0CL0AS3V : hm, fair point
U0CL0AS3V : yeah I suppose that doesn't work either
U2GTQM83A : The ideal solution would involve creating something like a `Decoder`. So that I say that an Http with `Decoder a` would need to have an `EmbedDecoder a`. This way I tie what is being decoded with what is being embedded. And then I need some function that could translate an `EmbedDecoder a` to a string.
U2GTQM83A : And some way I could compose `EmbedDecoder a` to create a `EmbedDecoder (Project a)`
U0CL0AS3V : okay, what about this? (in the status quo `type alias` version)
several functions, but using extensible records to avoid combinatorics:

for example:

```
{ a | owner : User String } -&gt; String

{ a | owner : User () } -&gt; String

{ a | contacts : List (ProjectContact ()) } -&gt; String
```

U2GTQM83A : Hmm.. I don't see how to use it.
U2GTQM83A : There is another point also. At the time of the Http, I probably don't have a value of type `Book a`. I'm not trying to decode the value itself. I need a constraint on the type. Like a Decoder.
U0CL0AS3V : `{ a | owner : User () } -> String` accepts a `ProjectStructure (User ()) company contact`
U0CL0AS3V : because `ProjectStructure (User ()) company contact` is a record which has a field called `owner` of type

`User ()`
U2GTQM83A : I want to make sure that if I am making an Http request to my `book` endpoint requesting a `Book Author`, this http request must ask for the author to be embedded.
U0CL0AS3V : I don't think it's (even theoretically) possible to be confident of that without tests :thinking_face:
U0CL0AS3V : in any language
U0CL0AS3V : like "the author is embedded" is business logic, which can be incorrect
U0CL0AS3V : I'm not even sure how you'd use a theorem prover to check that