```ts
export type Infer<FujiSchema> = FujiSchema extends Fuji<
  infer $RuleType,
  infer Value
>
  ? Value extends AnyShapeSchema
    ? InferRecord<Value>
    : Value extends Array<AnyShapeSchema>
    ? InferArrayOfRecords<Value>[]
    : RequiredType extends $RuleType
    ? NullableType extends $RuleType
      ? Value | null
      : Value
    : DefaultToType extends $RuleType
    ? Value
    : NullableType extends $RuleType
    ? Value | null
    : Value | undefined
  : never

type InferArrayOfRecords<Value extends Array<AnyShapeSchema>> = ...
type InferRecord<Shape extends Record<string, Fuji<any, any>>> = ...
type RequiredKeys<T extends AnyShapeSchema> = ...
type OptionalKeys<T extends AnyShapeSchema> = ...
```

<infer T>

```ts
const promise = Promise.resolve(42) // Promise<number>

type Await<A> = A extends Promise<infer B> ? B : never;

/**
  if A is typeof Promise<?>
  then take ? and save it's type to B
**/

type Result = Await<typeof promise> // number
```