# Introdução

A telefonia foi o primeiro grande passo para a popularização das telecomunicações, trazendo ao público em geral a capacidade de se comunicar a distâncias gigantescas. Além dos avanços tecnológicos em transmissão, modulação e criptografia de sinais que a telefonia proporcionou, as linhas telefônicas são um exemplo bastante ubíquo de máquina de estados finitos. Um exemplo exaustivamente testado durante mais de uma centena de anos de uso.

A proposta do trabalho é modelar a máquina de estados de uma linha telefônica simples: para efeitos de modelagem, considerou-se uma linha fixa trivial, que somente aceita e origina ligações. Uma vez modelada a linha em si, o objetivo final do trabalho é modelar uma central telefônica capaz de gerenciar duas ou mais dessas linhas telefônicas e garantir que chamadas possam ser completadas entre essas N linhas assinantes de serviço.

Muito embora a modelagem de cada linha em si seja trivial, o problema torna-se incrementalmente mais atrativo com o aumento do número de linhas gerenciadas, já que a quantidade de estados da central cresce exponencialmente em relação a combinação dos estados das linhas gerenciadas.

# Metodologia

Ficou decidido que o primeiro passo seria modelar uma linha telefônica em separado com todos os seus estados. Feito isso, pode-se extrair algumas propriedades relativas à linha telefônica em si, o que deveria ser trivial. Em seguida modelar-se-ia a central telefônica com duas linhas, verificando-se novas propriedades. Por fim, seria possível aumentar arbitrariamente o número de linhas telefônicas no modelo até o ponto em que fosse viável provar propriedades nos modelos gerados.

## Modelagem

Como previamente citado, decidiu-se por modelar uma linha telefônica padrão fixa, sem serviços extras de assinante. A máquina de estados resultante da modelagem é a seguinte:



Na figura estão modelados os estados da linha e nomeadas suas transições. Foram omitidos todos os loops para efeitos de legibilidade, porém é suficiente dizer que todos os estados, com exceção do estado de erro, podem se manter em loop enquanto as condições para a transição não se satisfizerem. Em especial, num mundo ideal, diversos destes loops possuem timeouts, que não foram implementados *a priori*.

Para a modelagem da central telefônica, três máquinas de estados semelhantes à descrita pelo diagrama acima foram acopladas num módulo SMV que representaria a central. Durante a implementação, diversos problemas surgiram, sobretudo devido à inexperiência na operação do software. Tais problemas estão descritos no Apêndice I: Notas do aluno.

## Implementação

A implementação foi feita utilizando a linguagem padrão da ferramenta NuSMV, tanto para a descrição da máquina de estados quanto para a definição das propriedades a serem provadas. O código SMV encontra-se no Apêndice II: Código.

Em especial, um dos problemas surgidos durante a implementação se deve à forma de controlar a transição de estados das linhas telefônicas. Uma vez que cada uma das linhas depende do estado das demais (uma linha B pode forçar uma mudança em uma linha A, quando tenta se conectar a ela), foi necessária a implementação de variáveis externas, chamadas no código de lineX\_connect, para monitorar a quem uma determinada linha está conectada ou tentando se conectar. Na prática, essas variáveis funcionam tanto como semáforos rudimentares quanto indicadores do estado de uma linha remota para efeitos de controle de estados da linha local.

Feito isso, cada transição de estados de uma linha depende não somente do estado atual, mas também da variável externa relacionada.

## Extração de Propriedades

As seguintes propriedades foram extraídas do modelo gerado:

|  |
| --- |
| Se A está conectado em B, seja ligando ou falando, B não deve estar conectado em C |
| SPEC AG !( (line0 in { CALL, TALK } & line0\_connect = LINE1) & line1\_connect = LINE2 )  SPEC AG !( (line0 in { CALL, TALK } & line0\_connect = LINE2) & line2\_connect = LINE1 )  SPEC AG !( (line1 in { CALL, TALK } & line1\_connect = LINE2) & line2\_connect = LINE0 ) |

|  |
| --- |
| Se A está falando com B, B deve estar também falando |
| SPEC AG !( (line0 = TALK & line0\_connect = LINE1) & line1 != TALK )  SPEC AG !( (line0 = TALK & line0\_connect = LINE2) & line2 != TALK )  SPEC AG !( (line1 = TALK & line1\_connect = LINE0) & line0 != TALK )  SPEC AG !( (line1 = TALK & line1\_connect = LINE2) & line2 != TALK )  SPEC AG !( (line2 = TALK & line2\_connect = LINE0) & line0 != TALK )  SPEC AG !( (line2 = TALK & line2\_connect = LINE1) & line1 != TALK ) |

|  |
| --- |
| Se A está tocando, B ou C devem estar conectados a A |
| SPEC AG !( line0 = RING & !(line1\_connect = LINE0 | line2\_connect = LINE0) )  SPEC AG !( line1 = RING & !(line0\_connect = LINE1 | line2\_connect = LINE1) )  SPEC AG !( line2 = RING & !(line0\_connect = LINE2 | line1\_connect = LINE2) ) |

|  |
| --- |
| Se A está ligando para B, B deve estar esperando ou tocando |
| SPEC AG !( (line0 = CALL & line0\_connect = LINE1) & !(line1 in { WAIT, RING }) )  SPEC AG !( (line0 = CALL & line0\_connect = LINE2) & !(line2 in { WAIT, RING }) )  SPEC AG !( (line1 = CALL & line1\_connect = LINE0) & !(line0 in { WAIT, RING }) )  SPEC AG !( (line1 = CALL & line1\_connect = LINE2) & !(line2 in { WAIT, RING }) )  SPEC AG !( (line2 = CALL & line2\_connect = LINE0) & !(line0 in { WAIT, RING }) )  SPEC AG !( (line2 = CALL & line2\_connect = LINE1) & !(line1 in { WAIT, RING }) ) |

|  |
| --- |
| Sempre é verdade que, se A está ligando para B, fatalmente eles conversarão |
| SPEC AG ( (line0 = CALL & line0\_connect = line1 ) -> AF (line1\_connect = LINE0 & line0 = TALK & line1 = TALK) )  SPEC AG ( (line0 = CALL & line0\_connect = line2 ) -> AF (line2\_connect = LINE0 & line0 = TALK & line2 = TALK) )  SPEC AG ( (line1 = CALL & line1\_connect = line0 ) -> AF (line0\_connect = LINE0 & line0 = TALK & line0 = TALK) )  SPEC AG ( (line1 = CALL & line1\_connect = line2 ) -> AF (line2\_connect = LINE0 & line0 = TALK & line2 = TALK) )  SPEC AG ( (line2 = CALL & line2\_connect = line0 ) -> AF (line0\_connect = LINE0 & line0 = TALK & line0 = TALK) )  SPEC AG ( (line2 = CALL & line2\_connect = line1 ) -> AF (line1\_connect = LINE0 & line0 = TALK & line1 = TALK) ) |

# Resultados

Embora bastante simplificado, o modelo gerado parece representar, de forma eficiente, uma central telefônica com três assinantes. O modelo prevê todas as transições de estados necessárias, além de implementar o controle da central com relação à correlação entre essas transições entre diferentes linhas.

As propriedades extraídas não exercitam todas as possibilidades do sistema, mas tentam garantir as funcionalidades básicas da central. Dito isso, todas elas foram provadas corretamente pelo software NuSMV em tempos desprezíveis, configurando a plataforma como bastante eficiente para a prova de modelos com nível de complexidade similar ao implementado

# Apêndice I: Notas do Aluno

A ideia inicial era produzir o modelo utilizando a linguagem Verilog e o software JasperGold Apps, mais especificamente sua aplicação FPV (*Formal Proof Verification*) para este trabalho, mas o modelo mostrou-se mais complexo do que o tempo disponível permitia para esse caso. Em especial, não consegui modelar de forma satisfatória, em Verilog, as transições não obrigatórias de estado. Faltou-me descobrir uma forma de representar transições do tipo:

next(estado) := { possível\_estado\_a, possível\_estado\_b };

em que o próximo estado pode ser uma de várias opções.

Visto isso, refiz o modelo utilizando a linguagem do NuSMV, porém creio que a modelagem resultante está longe da ideal. Em especial não consegui, em tempo hábil, criar um módulo genérico o suficiente para representar uma linha telefônica e que fosse totalmente independente a ponto de ser um módulo SMV instanciável: o modelo repete as regras e propriedades para cada instância de linha telefônica.

# Apêndice **II**: Código

|  |
| --- |
| MODULE main  VAR  line0: { WAIT, CALL, RING, TALK, DIAL, ERR };  line0\_connect: { NONE, LINE1, LINE2 };  line1: { WAIT, CALL, RING, TALK, DIAL, ERR };  line1\_connect: { NONE, LINE0, LINE2 };  line2: { WAIT, CALL, RING, TALK, DIAL, ERR };  line2\_connect: { NONE, LINE0, LINE1 };  ASSIGN  init(line0\_connect) := NONE;  next(line0\_connect) := case  line0\_connect = NONE: { LINE1, LINE2 };  line0\_connect = LINE1 & line0 in { WAIT, DIAL, CALL, RING } : LINE1;  line0\_connect = LINE2 & line0 in { WAIT, DIAL, CALL, RING }: LINE2;  line0\_connect = LINE1 & line0 = TALK: { NONE, LINE1 };  line0\_connect = LINE2 & line0 = TALK: { NONE, LINE2 };  TRUE: NONE;  esac;  init(line1\_connect) := NONE;  next(line1\_connect) := case  line1\_connect = NONE: { LINE0, LINE2 };  line1\_connect = LINE0 & line0 in { WAIT, DIAL, CALL, RING }: LINE0;  line1\_connect = LINE2 & line0 in { WAIT, DIAL, CALL, RING }: LINE2;  line1\_connect = LINE0 & line0 = TALK: { NONE, LINE0 };  line1\_connect = LINE2 & line0 = TALK: { NONE, LINE2 };  TRUE: NONE;  esac;  init(line2\_connect) := NONE;  next(line2\_connect) := case  line2\_connect = NONE: { LINE1, LINE0 };  line2\_connect = LINE1 & line0 in { WAIT, DIAL, CALL, RING }: LINE1;  line2\_connect = LINE0 & line0 in { WAIT, DIAL, CALL, RING }: LINE0;  line2\_connect = LINE1 & line0 = TALK: { NONE, LINE1 };  line2\_connect = LINE0 & line0 = TALK: { NONE, LINE0 };  TRUE: NONE;  esac;  init(line0) := WAIT;  next(line0) := case  -- Se estou em erro somente posso voltar pro WAIT  line0 = ERR: WAIT;  -- Se estou em wait e tento me conectar a outra linha, vou pra DIAL  -- Se estou em wait e alguem conecta vou para RING  line0 = WAIT & line0\_connect != NONE: DIAL;  line0 = WAIT & line1\_connect = LINE0: RING;  line0 = WAIT & line2\_connect = LINE0: RING;  line0 = WAIT & line0\_connect = NONE: WAIT;  -- Se eu estou falando e desconecto, volto pra WAIT  -- Se estamos conversando e a linha remota desliga, volto pro WAIT  line0 = TALK & line0\_connect = NONE: WAIT;  line0 = TALK & line0\_connect != NONE: TALK;  line0 = TALK & line0\_connect = LINE1 & line1\_connect = NONE: WAIT;  line0 = TALK & line0\_connect = LINE2 & line2\_connect = NONE: WAIT;  -- Se estou tocando e me conecto, falamos (TALK)  line0 = RING & line0\_connect != NONE: TALK;  line0 = RING & line0\_connect = NONE: RING;  -- Se estou discando e a linha remota esta livre, passa a tocar  line0 = DIAL & line0\_connect = LINE1 & line1\_connect = NONE: CALL;  line0 = DIAL & line0\_connect = LINE1 & line1\_connect != NONE: ERR;  line0 = DIAL & line0\_connect = LINE2 & line2\_connect = NONE: CALL;  line0 = DIAL & line0\_connect = LINE2 & line2\_connect != NONE: ERR;  -- Se esta tocando e a linha remota conecta em mim, conversamos  line0 = CALL & line0\_connect = LINE1 & line1\_connect = LINE0: TALK;  line0 = CALL & line0\_connect = LINE1 & line1\_connect = NONE: CALL;  line0 = CALL & line0\_connect = LINE2 & line2\_connect = LINE0: TALK;  line0 = CALL & line0\_connect = LINE2 & line2\_connect = NONE: CALL;  TRUE: ERR;  esac;  init(line1) := WAIT;  next(line1) := case  -- Se estou em erro somente posso voltar pro WAIT  line1 = ERR: WAIT;  -- Se estou em wait e tento me conectar a outra linha, vou pra DIAL  -- Se estou em wait e alguem conecta vou para RING  line1 = WAIT & line1\_connect != NONE: DIAL;  line1 = WAIT & line0\_connect = LINE1: RING;  line1 = WAIT & line2\_connect = LINE1: RING;  line1 = WAIT & line1\_connect = NONE: WAIT;  -- Se eu estou falando e desconecto, volto pra WAIT  -- Se estamos conversando e a linha remota desliga, volto pro WAIT  line1 = TALK & line1\_connect = NONE: WAIT;  line1 = TALK & line1\_connect != NONE: TALK;  line1 = TALK & line1\_connect = LINE0 & line0\_connect = NONE: WAIT;  line1 = TALK & line1\_connect = LINE2 & line2\_connect = NONE: WAIT;  -- Se estou tocando e me conecto, falamos (TALK)  line1 = RING & line1\_connect != NONE: TALK;  line1 = RING & line1\_connect = NONE: RING;  -- Se estou discando e a linha remota esta livre, passa a tocar  line1 = DIAL & line1\_connect = LINE0 & line0\_connect = NONE: CALL;  line1 = DIAL & line1\_connect = LINE0 & line0\_connect != NONE: ERR;  line1 = DIAL & line1\_connect = LINE2 & line2\_connect = NONE: CALL;  line1 = DIAL & line1\_connect = LINE2 & line2\_connect != NONE: ERR;  -- Se esta tocando e a linha remota conecta em mim, conversamos  line1 = CALL & line1\_connect = LINE0 & line0\_connect = LINE1: TALK;  line1 = CALL & line1\_connect = LINE0 & line0\_connect = NONE: CALL;  line1 = CALL & line1\_connect = LINE2 & line2\_connect = LINE1: TALK;  line1 = CALL & line1\_connect = LINE2 & line2\_connect = NONE: CALL;  TRUE: ERR;  esac;  init(line2) := WAIT;  next(line2) := case  -- Se estou em erro somente posso voltar pro WAIT  line2 = ERR: WAIT;  -- Se estou em wait e tento me conectar a outra linha, vou pra DIAL  -- Se estou em wait e alguem conecta vou para RING  line2 = WAIT & line2\_connect != NONE: DIAL;  line2 = WAIT & line0\_connect = LINE2: RING;  line2 = WAIT & line1\_connect = LINE2: RING;  line2 = WAIT & line2\_connect = NONE: WAIT;  -- Se eu estou falando e desconecto, volto pra WAIT  -- Se estamos conversando e a linha remota desliga, volto pro WAIT  line2 = TALK & line2\_connect = NONE: WAIT;  line2 = TALK & line2\_connect != NONE: TALK;  line2 = TALK & line2\_connect = LINE0 & line0\_connect = NONE: WAIT;  line2 = TALK & line2\_connect = LINE1 & line1\_connect = NONE: WAIT;  -- Se estou tocando e me conecto, falamos (TALK)  line2 = RING & line2\_connect != NONE: TALK;  line2 = RING & line2\_connect = NONE: RING;  -- Se estou discando e a linha remota esta livre, passa a tocar  line2 = DIAL & line2\_connect = LINE0 & line0\_connect = NONE: CALL;  line2 = DIAL & line2\_connect = LINE0 & line0\_connect != NONE: ERR;  line2 = DIAL & line2\_connect = LINE1 & line1\_connect = NONE: CALL;  line2 = DIAL & line2\_connect = LINE1 & line1\_connect != NONE: ERR;  -- Se esta tocando e a linha remota conecta em mim, conversamos  line2 = CALL & line2\_connect = LINE0 & line0\_connect = LINE2: TALK;  line2 = CALL & line2\_connect = LINE0 & line0\_connect = NONE: CALL;  line2 = CALL & line2\_connect = LINE1 & line1\_connect = LINE2: TALK;  line2 = CALL & line2\_connect = LINE1 & line1\_connect = NONE: CALL;  TRUE: ERR;  esac;  -- Se A esta conectado em B, B deve estar conectado em A ou nao estar conectado  SPEC AG !( (line0 in { CALL, TALK } & line0\_connect = LINE1) & line1\_connect = LINE2 )  SPEC AG !( (line0 in { CALL, TALK } & line0\_connect = LINE2) & line2\_connect = LINE1 )  SPEC AG !( (line1 in { CALL, TALK } & line1\_connect = LINE2) & line2\_connect = LINE0 )  -- Se A esta falando com B, B deve estar conectado a A  SPEC AG !( (line0 = TALK & line0\_connect = LINE1) & line1 != TALK )  SPEC AG !( (line0 = TALK & line0\_connect = LINE2) & line2 != TALK )  SPEC AG !( (line1 = TALK & line1\_connect = LINE0) & line0 != TALK )  SPEC AG !( (line1 = TALK & line1\_connect = LINE2) & line2 != TALK )  SPEC AG !( (line2 = TALK & line2\_connect = LINE0) & line0 != TALK )  SPEC AG !( (line2 = TALK & line2\_connect = LINE1) & line1 != TALK )  -- Se A esta tocando, B ou C precisam estar conectados a A  SPEC AG !( line0 = RING & !(line1\_connect = LINE0 | line2\_connect = LINE0) )  SPEC AG !( line1 = RING & !(line0\_connect = LINE1 | line2\_connect = LINE1) )  SPEC AG !( line2 = RING & !(line0\_connect = LINE2 | line1\_connect = LINE2) )  -- Se A esta ligando para B, B deve estar em WAIT ou RINGING  SPEC AG !( (line0 = CALL & line0\_connect = LINE1) & !(line1 in { WAIT, RING }) )  SPEC AG !( (line0 = CALL & line0\_connect = LINE2) & !(line2 in { WAIT, RING }) )  SPEC AG !( (line1 = CALL & line1\_connect = LINE0) & !(line0 in { WAIT, RING }) )  SPEC AG !( (line1 = CALL & line1\_connect = LINE2) & !(line2 in { WAIT, RING }) )  SPEC AG !( (line2 = CALL & line2\_connect = LINE0) & !(line0 in { WAIT, RING }) )  SPEC AG !( (line2 = CALL & line2\_connect = LINE1) & !(line1 in { WAIT, RING }) )  -- Sempre e verdade que se A esta ligando para B, fatalmente eles conversarao  SPEC AG ( (line0 = CALL & line0\_connect = line1 ) -> AF (line1\_connect = LINE0 & line0 = TALK & line1 = TALK) )  SPEC AG ( (line0 = CALL & line0\_connect = line2 ) -> AF (line2\_connect = LINE0 & line0 = TALK & line2 = TALK) )  SPEC AG ( (line1 = CALL & line1\_connect = line0 ) -> AF (line0\_connect = LINE0 & line0 = TALK & line0 = TALK) )  SPEC AG ( (line1 = CALL & line1\_connect = line2 ) -> AF (line2\_connect = LINE0 & line0 = TALK & line2 = TALK) )  SPEC AG ( (line2 = CALL & line2\_connect = line0 ) -> AF (line0\_connect = LINE0 & line0 = TALK & line0 = TALK) )  SPEC AG ( (line2 = CALL & line2\_connect = line1 ) -> AF (line1\_connect = LINE0 & line0 = TALK & line1 = TALK) ) |