RAILKERNEL

Vrij rijden: onderzoeksnotities en ontwerpregels

Status: werkdocument voor de verdere ontwikkeling van RailKernel 12.01.

Dit document vertaalt wetenschappelijke publicaties over spoorwegroutering, deadlockpreventie, resource-reservering en multi-agent planning naar concrete ontwerpregels voor RailKernels **Vrij rijden**. Het is geen beschrijving van de huidige implementatie en ook geen besluit om een compleet academisch algoritme over te nemen. Het maakt zichtbaar welke onderdelen direct bruikbaar zijn, welke aannames niet bij een modelbaan passen en waar de huidige implementatie nog conservatiever of juist veiliger moet worden.

1. RailKernel-model waarop dit document voortbouwt

RailKernel beschikt al over onderdelen die in veel publicaties eerst nog als abstract model moeten worden ingevoerd:

- een geometrisch baanmodel met lengtes, aansluitingen en rijrichtingen;
- blokken en accessories als exclusief reserveerbare resources;
- feedbacks als fysieke positie-events;
- gegenereerde, gerichte A-B-C-Triplets;
- een vooraf berekende fysieke B-C-corridor met wisselstanden;
- een geforceerde vervolgreeks tot een splitsing, eindpunt of cyclus;
- gezamenlijke kandidaatkeuze voor meerdere vertrekkende treinen;
- een geometrisch berekende vrijgavetabel per feedback;
- treinlengte, detectielengte, snelheidscalibratie en remwegcalibratie;
- debugrapporten die een keuze en reservering verklaarbaar maken.

De kern van Vrij rijden is geen route naar één vooraf vastgelegde eindbestemming. Bij ieder beslismoment kiest RailKernel een veilige, lokale voortgang. Een splitsing, passeerplaats of ander bewezen veilig punt kan daarbij als tijdelijke bestemming dienen.

2. Begrippen vertalen naar RailKernel

Literatuur	RailKernel
resource / track segment	blok, wissel of andere exclusieve accessory
node / section	gericht blok of fysieke corridor
movement	gekozen Triplet en zijn B-C-corridor
successor	blok C van het gekozen Triplet
path	reeks aansluitende Triplets
buffer	geforceerde vervolgreeks tot een bewezen uitweg
safe place / endpoint	plaats waar een trein kan wachten zonder noodzakelijke doorgang te blokkeren
train process	trein plus zijn huidige en mogelijk volgende resourcebehoefte
conflict	twee claims die niet gelijktijdig uitgevoerd kunnen worden
release event	feedbackevent waarbij de treinachterkant een resource aantoonbaar heeft verlaten
stop checking point	punt waar nog veilig gestopt of afgeremd kan worden voor een niet-beschikbare opvolger
safe state	vanuit deze toestand bestaat een uitvoeringsvolgorde waarin alle betrokken treinen weer voortgang kunnen maken

Een belangrijk terminologisch verschil is dat **niet geblokkeerd** niet hetzelfde is als **veilig**. Een toestand kan nu nog beweging toelaten, terwijl iedere mogelijke vervolgkeuze later tot deadlock leidt.

3. Niet-onderhandelbare ontwerpregels

3.1 Veiligheid en voorkeur zijn twee verschillende lagen

Eerst wordt bepaald welke bewegingen veilig en fysiek geldig zijn. Pas daarna mogen prioriteit, gewicht, spreiding, willekeur, afstand en eerder gebruik een keuze maken tussen de toegestane kandidaten.

Een kandidaat met een hoge prioriteit mag nooit een onveilige kandidaat worden door het scoresysteem. Omgekeerd mag een veilige kandidaat niet stilzwijgend verdwijnen doordat een voorkeursscore als validatieregel wordt gebruikt.

3.2 Iedere beweging krijgt een controleerbaar veiligheidscertificaat

Voor een voorgenomen Triplet moet vóór vertrek vaststaan:

- welke resources onmiddellijk worden geclaimd;
- welke resources fysiek worden geschakeld;
- welke geforceerde vervolgstappen zijn onderzocht;
- welke vrije uitweg of andere trein voortgang garandeert;
- van welke andere treinen de veiligheid afhankelijk is;
- dat deze afhankelijkheden geen gesloten wachtcyclus vormen;
- op welke feedbacks resources weer worden vrijgegeven.

Dit is conceptueel een SafetyCertificate; het hoeft niet gepersisteerd te worden en kan bij iedere beslissing opnieuw uit de actuele toestand ontstaan.

3.3 Dezelfde rijrichting is nuttige informatie, geen bewijs

Een bezet blok verderop met een bekende trein in dezelfde richting hoeft de keuze niet direct te blokkeren. Die trein kan de benodigde uitweg vrijmaken. Maar dat mag alleen worden aangenomen wanneer voor die trein recursief een geldige voortgang naar een vrije uitweg kan worden aangetoond.

Daarom geldt:

- dezelfde richting plus aantoonbare uitweg: bruikbare voortgangsgetuige;
- dezelfde richting zonder aantoonbare uitweg: nog niet veilig;
- tegengestelde richting: conflict, tenzij een expliciete passeer- of omkeerstrategie dit oplost;
- onbekende trein, eigenaar of rijrichting: conservatief blokkeren;
- een cyclus in de keten van voortgangsafhankelijkheden: onveilig.

Dit verfijnt de huidige gedachte “de trein voor mij zal het vuile werk doen”. Dat is vaak waar, maar moet door de hele keten tot een echte uitweg bewezen worden.

3.4 Reserveren is atomair

Een beweging wordt pas gecommit wanneer alle onmiddellijk benodigde blokken en accessories samen gereserveerd kunnen worden. Bij mislukking of annulering worden alle voorlopige claims teruggedraaid. Er mag geen half gereserveerde corridor overblijven.

3.5 Vrijgave volgt de treinachterkant

Een resource mag niet worden vrijgegeven zolang enig deel van de trein erop staat. De primaire RailKernel-methode blijft daarom de geometrisch berekende vrijgavetabel:

- feedback bezet is een betrouwbare fysieke positie-anchor;
- de afstand vanaf het resource-einde wordt vooraf uit de geometrie berekend;
- treinlengte en detectie-overhang bepalen of de achterkant vrij is;
- de laatste feedback in C is de veiligheidsvangnet-trigger die resterende entry-resources en blok A vrijgeeft.

Tijd- of snelheidsgebaseerde vrijgave mag alleen als aanvullende methode worden gebruikt. Wanneer snelheid verandert, moeten nog geplande vrijgavemomenten dan opnieuw worden berekend. Een feedback-geometrische vrijgave heeft dit probleem niet.

3.6 Een splitsing is niet automatisch een veilige plaats

Een “eerste blok met meerdere vervolgkeuzes” is een nuttige tijdelijke horizon, maar alleen een veilige wachtplaats wanneer:

- de trein er fysiek kan staan zonder de achterliggende corridor te blokkeren;
- minstens één latere voortgangsmogelijkheid werkelijk beschikbaar kan worden;
- de plaats geen essentieel doorgangspunt voor andere treinen afsluit;
- de toegestane richting en treinlengte kloppen.

Veilige plaatsen zijn dus trein- en richtingafhankelijk.

4. Wat de publicaties concreet leren

4.1 Lu, Dessouky en Leachman: Modeling Train Movements Through Complex Rail Networks

Dit artikel sluit het dichtst aan bij RailKernel. Het model gebruikt exclusieve spoorsegmenten en junction-resources, legt fysieke beweging vast en maakt een duidelijk onderscheid tussen een toestand zonder actuele botsing en een veilige toestand.

Direct toepasbaar:

- reserveer zowel opvolgend spoor als verbindende junction-resources;
- beslis vóór het punt waarop een trein niet meer veilig voor de opvolger kan stoppen;
- gebruik een buffer voorbij de onmiddellijke opvolger om te voorkomen dat een trein een onveilig punt binnenrijdt;
- simuleer een mogelijke beweging en onderzoek daarna welke treinen hun route kunnen voltooien en resources kunnen teruggeven;
- beschouw gelijkgerichte treinen als positieve doorstroming in de tie-breaking, niet automatisch als obstakel;

- herbereken tijdgebaseerde resource-free-events wanneer een trein versnelt of vertraagt.

Algorithm 4.1 vereist een volledig vrije weg tot de eindbestemming en is voor RailKernel veel te conservatief. Algorithm 4.2 gebruikt een kortere buffer, maar claimt die buffer daadwerkelijk. RailKernel wil scherper rijden en kan daarom het verschil expliciet maken tussen:

1. onmiddellijk geclaimde resources; en
2. een bewezen maar nog niet volledig geclaimde voortgangsketen.

Dat tweede is krachtiger maar moet met afhankelijkheden en cyclusdetectie worden beveiligd.

4.2 Dessouky e.a.: Algorithms for Scheduling Freight Trains in Complex Rail Networks

De Greedy-methode uit dit rapport is een snelle, lokale afleiding van dezelfde bufferbenadering. De rekentijd is gering, maar de oplossing heeft meer vertraging dan duurdere, verder vooruitkijkende methoden.

Voor RailKernel betekent dit:

- een lokale online heuristiek is verantwoord voor ieder feedbackmoment;
- noem haar niet globaal optimaal;
- laat een begrensde, duurdere zoekslag alleen draaien wanneer kandidaten elkaar beïnvloeden of de lokale keuze geen veilige voortgang vindt;
- houd de snelle standaardroute en de veiligheidsfallback apart.

4.3 Simon, Jaumard en Le: Deadlock Avoidance and Detection in Railway Simulation Systems

Deze methode bouwt dynamisch een geordende reserveringsketen per sectie. Een trein mag pas rijden wanneer zijn reservering bevestigd is. Bij een voorliggende trein in dezelfde richting wordt niet alleen aangenomen dat het goedkomt: de reserveringsprocedure voor die trein wordt recursief uitgevoerd.

Direct toepasbaar:

- modelleer een voortgangsafhankelijkheid trein A wacht op voortgang van B;
- bewijs de keten recursief en detecteer herhaling;
- geef een trein pas vertrektoestemming nadat het gehele bewijs bevestigd is;
- maak wachtvolgorde en resource-eigenaarschap zichtbaar in debugoutput.

Beperking: het artikel veronderstelt enkelspoorsecties met eenvoudige passeerplaatsen en vaste eindbestemmingen. RailKernel heeft rijkere topologie en wisselende tijdelijke doelen, dus het algoritme is geen één-op-één implementatie.

4.4 Cui e.a.: Banker's Algorithm met feasible resources

De klassieke Banker's Algorithm controleert of alle actieve processen in een mogelijke volgorde hun maximale resourcebehoefte kunnen krijgen, voltooien en daarna hun resources teruggeven. Dit is veilig maar kan te veel veilige aanvragen weigeren. Cui beperkt de onderzochte behoefte tot bereikbare, relevante downstreamresources en vermindert daarmee zulke false positives sterk.

Voor RailKernel:

- gebruik niet “alle resources tot een uiteindelijke bestemming” als maximale behoefte bij Vrij rijden;
- gebruik de geforceerde Tripletketen tot een echte uitweg als begrensde resourcebehoefte;
- voer een virtuele allocatie uit en verwijder iteratief treinen die aantoonbaar kunnen voltooien;
- keur de toestand af wanneer geen enkele resterende trein verwijderd kan worden.

Dit past goed bij een toekomstige SafetyCertificate-bouwer.

4.5 Dal Sasso e.a.: The Tick Formulation for Deadlock Detection and Avoidance

Deze benadering deelt beweging op in discrete stappen (“ticks”) en formuleert resourcevolgorde en deadlockvrijheid als een 0/1-model. Het sterke idee is niet dat RailKernel tijdens iedere feedback een volledige optimizer moet starten, maar dat een voorgenomen set bewegingen als één gezamenlijk probleem kan worden gecontroleerd.

Praktische inzet:

- gebruik een gezamenlijke dispatch-ronde voor treinen die tegelijk klaarstaan;
- zoek de grootste conflictvrije subset van vertrekkers;
- bewaar de reden waarom een trein deze ronde wacht;
- overweeg later een offline/diagnostische solver, niet in de normale GUI-loop.

4.6 Doshi e.a.: lineaire deadlockdetectie met een next-stop graph

De auteurs bouwen een gerichte graaf met voor iedere trein alleen de huidige en volgende resource. Een resource is rood wanneer nog capaciteit beschikbaar is en zwart wanneer hij vol is. Onder de aanname dat iedere resource minstens twee sporen capaciteit heeft, is de toestand veilig precies wanneer ieder zwart punt een gericht pad naar een rood punt heeft.

De interessante RailKernel-les is de **voortgangsgraaf**: met zeer weinig informatie kunnen afhankelijkheidsketens en gesloten zwarte cycli zichtbaar worden. De bewezen noodzakelijke-en-voldoende regel geldt echter niet voor een modelbaan waar vrijwel ieder blok capaciteit één heeft. Gebruik hem dus als:

- inspiratie voor een lineaire diagnosegraaf;
- snelle waarschuwing en debugvisualisatie;
- nooit als algemeen veiligheidsbewijs voor RailKernel.

4.7 Giua, Seatzu en Fanti: Petri-netten en supervisory control

Deze publicaties scheiden logische besturing van prestatiebesturing. Plaatsen representeren resources, tokens representeren treinen en monitors verhinderen toestanden die botsing of verlies van liveness veroorzaken.

Voor RailKernel is vooral de architectuur waardevol:

- harde veiligheid en liveness zitten onder de beleidskeuze;
- treinsoort, richting en toestand kunnen als “kleur” van een token worden beschouwd;
- structurele layoutregels kunnen offline worden gecompileerd;
- een aparte layoutcontrole kan globale invariant- en cyclusproblemen melden.

Een volledige Petri-net-engine in RailKernel is nu niet proportioneel. Een latere offline verifier voor Triplets en reserveringsafhankelijkheden kan wel op deze theorie worden gebaseerd.

4.8 D'Ariano e.a.: reordering en lokale rerouting

De ROMA-benadering behandelt resourcevolgorde en routekeuze als samenhangende, maar afzonderlijke problemen. Eerst worden conflicterende treinvolgordes gekozen; vervolgens wordt lokaal een alternatieve route gezocht en opnieuw geëvalueerd.

Voor RailKernel:

- houd “welke trein krijgt de resource eerst?” los van “welk Triplet kiest de trein?”;
- probeer bij een conflict een lokale alternatieve Tripletkeuze voordat een hele planning wordt verworpen;
- beperk herplanning tot het betrokken conflictgebied;
- maak de gekozen volgorde onderdeel van het debugrapport.

De publicatie is dienstregelingsgericht en veronderstelt een bekende horizon; Vrij rijden heeft juist steeds tijdelijke bestemmingen.

4.9 Carey: route-, lijn- en perronkeuze

Carey laat zien dat lijn-, perron- en routealternatieven expliciet als keuzes naast tijd- en headwayvoorwaarden gemodelleerd kunnen worden. Dat ondersteunt RailKernels keuze om meerdere Triplets naar fysiek verschillende corridors niet te vroeg samen te voegen.

Praktisch:

- een bestemmingblok alleen is geen volledige kandidaatidentiteit;

- fysieke corridor, wisselstanden, inrijzijde en uitrijzijde horen bij de sleutel;
- perron- of stationvoorkeur is beleid, niet fysieke geldigheid.

4.10 Lifelong MAPF en Token Passing

Gewone Multi-Agent Path Finding veronderstelt vaste doelen en eindigt wanneer iedere agent daar aankomt. Lifelong MAPF behandelt juist een doorgaande stroom van nieuwe opdrachten. Dat lijkt conceptueel meer op Vrij rijden.

De belangrijkste les is het begrip **endpoint**: een agent mag langdurig rusten op een plaats waar hij andere agenten niet blokkeert. Token Passing bewaart een gesynchroniseerd geheel van alle geplande paden; één agent tegelijk past zijn pad conflictvrij aan.

Voor RailKernel:

- behandel een veilige wachtplaats als expliciet layoutbegrip;
- een debug-ronde lijkt op een token: de gezamenlijke kandidaatset wordt eerst consistent gemaakt en daarna atomair gestart;
- parkeer een trein zonder geldige taak niet op een doorgaande bottleneck;
- de formele garantie voor “well-formed” magazijnen geldt niet automatisch voor spoorbanen, maar de endpointvoorwaarden zijn een bruikbare layoutcheck.

4.11 Conflict-Based Search (CBS)

CBS plant eerst voor iedere agent afzonderlijk. Wanneer twee plannen botsen, splitst de hoge zoeklaag het probleem in twee takken: in de ene mag trein A de conflictresource niet op dat moment gebruiken, in de andere trein B niet. Alleen het getroffen individuele pad wordt opnieuw berekend.

Voor RailKernel is een volledige tijdstap-CBS te zwaar en te kunstmatig zolang we met echte, variabele treinsnelheden en feedbackevents rijden. Een begrensde variant is wel aantrekkelijk voor gezamenlijke vertrekkeuze:

- begin met de beste kandidaat per trein;
- detecteer gedeelde blokken/accessories en onverenigbare afhankelijkheden;
- maak per conflict twee beperkte keuzetakken;
- herplan alleen de betrokken trein;
- stop na een tijd-, diepte- of kandidaatlimiet en kies dan de grootste veilig uitvoerbare subset.

4.12 Inglenook als operationele testcase

Inglenook is geen deadlockalgoritme, maar een compacte en waardevolle testcase voor rangeren:

- doodlopende sporen en omkeren;
- beperkte spoorcapaciteit;
- wagonvolgorde en samenstelling;
- een tijdelijk doel in plaats van een vaste eindbestemming;
- voorkomen dat een locomotief zichzelf insluit.

Het hoort later in de regressieset voor Vrij rijden en rangeren.

5. Aanbevolen beslispijplijn

Fase 0: offline compilatie bij Check Layout

1. Genereer gerichte Triplets en fysieke corridors.
2. Leg conflicterende resourceclaims vast.
3. Genereer de geometrische vrijgavetabellen.
4. Classificeer potentiële veilige wachtplaatsen per richting en treinlengte.
5. Rapporteer onbruikbare blokken, ontbrekende ontsnappingsmogelijkheden en structurele afhankelijkheidscycli.

Fase 1: kandidaten per trein

1. Zoek de Triplets die bij positie en rijrichting passen.
2. Filter op fysieke geldigheid, block direction, usable, treintype en lengte.
3. Verwijder onmiddellijke terugkeer, tenzij een expliciete omkeerbeweging wordt uitgevoerd.
4. Bouw per kandidaat de geforceerde vervolgreeks.

Fase 2: veiligheidscertificaat

1. Bepaal de onmiddellijke block- en accessoryclaims.
2. Loop de geforceerde reeks vooruit tot een bewezen veilige plaats.
3. Komt een vrije uitweg voor: bewijs geslaagd.
4. Komt een bekende, gelijkgerichte trein voor: bouw recursief diens bewijs.
5. Komt tegengestelde of onbekende bezetting voor: kandidaat blokkeren.
6. Komt dezelfde trein/afhankelijkheid opnieuw voor: cyclus, kandidaat blokkeren.
7. Bewaar het resultaat als verklaarbaar certificaat.

Fase 3: gezamenlijke selectie

1. Neem alle treinen die nu mogen proberen te vertrekken.
2. Kies aanvankelijk hun beste veilige kandidaat.
3. Detecteer onderlinge claim- en afhankelijkheidsconflicten.
4. Probeer begrensd alternatieven, geïnspireerd door CBS/backtracking.
5. Maximaliseer eerst het aantal veilige vertrekkers; optimaliseer daarna beleid.
6. Treinen zonder keuze wachten stil en proberen bij een relevante statechange opnieuw.

Fase 4: commit en vertrek

1. Reserveer alle onmiddellijke claims atomair.
2. Controleer dat het veiligheidscertificaat nog op dezelfde toestand berust.
3. Schakel accessories.
4. Bepaal snelheid en rem-/approachplan.
5. Geef het rijcommando.

Fase 5: positie, vrijgave en herplanning

1. Verwerk ieder feedbackevent als fysieke positie-anchor.
2. Pas de vooraf berekende vrijgavetabel toe op de treinachterkant.
3. Geef nooit een resource vrij waarop de trein nog staat.
4. Herplan zodra de beslisfeedback is bereikt of een afhankelijkheid wijzigt.
5. Herbereken eventuele tijdgebaseerde events na iedere snelheidswijziging.

6. Voorgestelde interne modellen

Deze namen zijn voorstellen, geen opdracht om ze direct allemaal te bouwen.

MovementCandidate

- train GUID;
- Triplet stable key;
- fysieke corridor;
- onmiddellijke blockclaims;
- onmiddellijke accessoryclaims;
- beleidskenmerken zoals prioriteit, gewicht en gebruiksteller;
- berekend SafetyCertificate.

SafetyCertificate

- toestand/revisie waarop het bewijs is gebaseerd;
- directe claims;
- geforceerde vervolgreeks;
- gevonden safe place;
- voortgangsafhankelijkheden naar andere treinen;
- bezochte treinen en bewegingen voor cyclusdetectie;
- afwijsreden of geldig resultaat;
- mensleesbare bewijsstappen voor debug.

ProgressDependency

- wachtende trein;
- voorliggende trein;
- resource waarop de afhankelijkheid ontstond;
- verwachte richting;
- onderliggend certificaat;
- status: PROVEN, WAITING, INVALIDATED.

SafePlace

- block GUID en gerichte in-/uitzijde;
- bruikbare treinlengte;
- mag langdurig bezet blijven;
- blokkeert geen noodzakelijke doorgaande corridor;
- uitwegen en voorwaarden;
- reden van classificatie.

DispatchRound

- deelnemers;
- kandidaten per trein;
- gevonden conflicten;
- onderzochte alternatieven;
- gekozen maximale veilige subset;
- atomische commitstatus.

7. Tests die rechtstreeks uit de literatuur volgen

Veiligheid en liveness

1. Een vrije geforceerde corridor eindigt bij een geldige uitweg: kandidaat mag.
2. Een gelijkgerichte trein staat vooraan en heeft zelf een bewezen uitweg: achterliggende kandidaat mag afhankelijk zijn van die trein.
3. Een keten gelijkgerichte treinen eindigt zonder uitweg: kandidaat mag niet.
4. Afhankelijkheden A -> B -> C -> A: alle betrokken kandidaten blokkeren en de cyclus rapporteren.
5. Tegengestelde trein op geforceerde corridor: kandidaat blokkeren.
6. Onbekende eigenaar of richting: kandidaat conservatief blokkeren.
7. Huidig conflictvrij maar onvermijdelijke latere deadlock: beweging weigeren.

Gezamenlijke dispatch

8 Twee kandidaten zijn afzonderlijk veilig maar claimen dezelfde wissel: maximaal één vertrekt.

9 Eén trein heeft een conflictvrij alternatief: beide vertrekken via andere kandidaten.

10 Drie treinen, slechts twee gelijktijdig uitvoerbaar: kies twee en laat de derde zonder foutpopup wachten.

11 Annulering tijdens voorbereiding: geen enkele voorlopige claim blijft staan.

Vrijgave

12 De locomotief heeft de wissel verlaten maar de wagons niet: wissel blijft gereserveerd.

13 De treinachterkant passeert het berekende resource-einde: vrijgave volgt op de juiste feedback.

14 De laatste feedback in C wordt geraakt: resterende entry-resources en A worden door de safety fallback vrijgegeven.

15 De trein versnelt of vertraagt: geometrische release blijft correct.

16 Een tijdgebaseerde fallback bestaat: zijn geplande tijden worden na de snelheidswijziging opnieuw berekend.

Safe places en bijzondere topologie

17 Een korte trein kan veilig wachten, een langere blokkeert dezelfde splitsing: classificatie verschilt per treinlengte.

18 Een unusable blok, transfertable of verboden richting vormt de juiste terminale grens.

19 Een doodlopend spoor wordt alleen gekozen wanneer stoppen en later omkeren expliciet ondersteund en veilig zijn.

20 Tegen-geometrisch rijden door een BOTH-blok behoudt correcte entry-, exit- en targetfeedbackrichting.

Determinisme en uitlegbaarheid

21 Dezelfde layout, toestand en selectiepolicy geeft dezelfde keuze.

22 Iedere weigering noemt de eerste falende harde regel.

23 Ieder vertrek toont claims, afhankelijkheden, veilige uitweg en releaseplan.

8. Geprioriteerde implementatieroute

Nu zinvol

1 Introduceer een expliciet SafetyCertificate rond de bestaande forcedContinuationForTrain-logica.

2 Vervang de onvoorwaardelijke gelijkgerichte-doorlaatregel door recursief voortgangsbewijs met cyclusdetectie.

3 Scheid harde kandidaatvalidatie strikt van priority/balancing/weight.

4 Laat debugoutput het volledige veiligheidsbewijs en afhankelijkheidsketen tonen.

5 Classificeer veilige wachtplaatsen expliciet in Check Layout.

6 Maak gezamenlijke selectie begrensd conflictgestuurd: alleen betrokken treinen en kandidaten vertakken.

Daarna

7 Voeg doodlopend-spoor/omkeerbewegingen als apart bewegingstype toe.

8 Hergebruik de geometrische vrijgavetabellen in de gewone routing.

9 Voeg een offline liveness-audit toe voor Triplets en afhankelijkheidscycli.

10 Onderzoek een diagnostische Petri-net- of 0/1-verifier voor kleine layouts.

Voorlopig niet doen

- geen volledige buffer standaard vooraf reserveren; dit is veilig maar onnodig conservatief en strijdig met scherp rijden;
- geen volledige route naar een vaste eindbestemming eisen bij Vrij rijden;
- geen volledige CBS-, MIP- of Petri-net-solver in ieder feedbackevent starten;
- een vrije splitsing niet zonder treinlengte- en doorgangscontrole als safe place aannemen;
- dezelfde rijrichting niet als zelfstandig deadlockbewijs gebruiken;
- “nu geen conflict” niet verwarren met “toestand blijft oplosbaar”.

9. Open ontwerpvragen

1. Wanneer is een blok formeel een veilige wachtplaats voor een specifieke trein?
2. Is een nog niet geclaimde voortgangsbelofte voldoende, of moet een deel van de geforceerde corridor toch hard gereserveerd worden?
3. Hoe lang blijft een SafetyCertificate geldig en welke statechanges maken het ongeldig?
4. Hoe groot mag een gezamenlijke dispatch-ronde worden voordat begrensde CBS- zoekactie te duur wordt?
5. Hoe modelleren we een trein die meerdere blokken en accessories tegelijk beslaat in de afhankelijkheidsgraaf?
6. Welke plekken mogen als parkeerendpoint gelden zonder doorgaand verkeer te hinderen?
7. Hoe wordt omkeren op een doodlopend spoor één atomische, veilige operatie?
8. Moet een voortgangsbelofte zichtbaar worden als een ander soort reservering op canvas en in monitors?

10. Bronnen

1. Lu, Dessouky & Leachman, *Modeling Train Movements Through Complex Rail Networks*: https://ieor.berkeley.edu/wp-content/uploads/2019/10/TrainMovements.pdf
2. Dessouky e.a., *Algorithms for Scheduling Freight Trains in Complex Rail Networks*: https://www.metrans.org/assets/research/10-08_Dessouky_METRANS_final_report_0_0.pdf
3. Cui e.a., *Searching Feasible Resources to Reduce False-Positive Situations for Resolving Deadlocks with the Banker's Algorithm in Railway Simulation*: https://doi.org/10.1016/j.jrtpm.2017.05.001
4. Dal Sasso e.a., *The Tick Formulation for Deadlock Detection and Avoidance in Railways Traffic Control*: https://doi.org/10.1016/j.jrtpm.2021.100239
5. Simon, Jaumard & Le, *Deadlock Avoidance and Detection in Railway Simulation Systems*: https://cclab.pages.in2p3.fr/bertrand.simon/files/Papers/2014-SJL-JRC.pdf
6. D'Ariano e.a., *Reordering and Local Rerouting Strategies to Manage Train Traffic in Real Time*: https://doi.org/10.1287/trsc.1080.0247
7. Doshi e.a., *Linear-Time Optimal Deadlock Detection for Efficient Scheduling in Multi-Track Railway Networks*: https://www.ijcai.org/proceedings/2024/0641.pdf
8. Giua & Seatzu, *Modeling and Supervisory Control of Railway Networks Using Petri Nets*: https://www.alessandro-giua.it/UNICA/PAPERS/JOUR/08tase_draft.pdf
9. Fanti, Giua & Seatzu, *Monitor Design for Colored Petri Nets: An Application to Deadlock Prevention in Railway Networks*: https://doi.org/10.1016/j.conengprac.2006.02.007
10. Carey, *A Model and Strategy for Train Pathing with Choice of Lines, Platforms, and Routes*: https://doi.org/10.1016/0191-2615(94)90033-7
11. Ma e.a., *Lifelong Multi-Agent Path Finding for Online Pickup and Delivery Tasks*: https://jiaoyangli.me/files/2017-AAMAS.pdf
12. Sharon e.a., *Conflict-Based Search for Optimal Multi-Agent Path Finding*: https://ojs.aaai.org/index.php/AAAI/article/view/8140
13. Croella e.a., *Disruption Management in Railway Systems by Safe Place Assignment*: https://sintef.brage.unit.no/sintef-xmlui/bitstream/handle/11250/3012116/TS-SafePlacePostPrint.pdf?sequence=2
14. Wymann, *Inglenook Sidings Shunting Puzzle - Rules & Operation*: https://www.wymann.info/ShuntingPuzzles/Inglenook/inglenook-rules.html

11. Samenvattende ontwerpbeslissing

RailKernel hoeft geen bestaand algoritme letterlijk over te nemen. De meest passende combinatie is:

- Triplets en geometrische release als RailKernel-specifieke fysieke basis;
- bufferdenken en virtuele voltooiing uit de railway-deadlockliteratuur;
- recursieve voortgangsafhankelijkheden voor voorliggende gelijkgerichte treinen;
- expliciete safe places uit lifelong planning;
- begrensde conflictgestuurde zoekactie voor gelijktijdige vertrekkers;
- strikte scheiding tussen veiligheid en voorkeur;
- een verklaarbaar veiligheidscertificaat vóór iedere daadwerkelijke beweging.

Daarmee blijft Vrij rijden lokaal, snel en bestemmingloos, zonder genoegen te nemen met de zwakkere regel dat een beweging veilig zou zijn omdat er op dit moment nog niets botst.